Hybrid Image Synthesis

Sahil Sidheekh
2017CSB1104
Indian Institute Of Technology, Ropar

Abstract

This document demonstrates the observations and results of synthesizing hybrid images using **first principles**, in **python** .This work consists of two parts - Implementing an image filter applier using convolution(correlation) and then using this to filter out low frequency and high frequency components of an image which are later combined to form the hybrid image.

1 Introduction

Convolution of images with kernels in-order to enhance image properties as per user needs is a popular approach used in image processing. It is a form of linear filtering in which the value of a pixel also depends on its neighbour pixels. Depending on the values of the kernel pixels and its size, the convolution results in different features of the image being enhanced. It is also used widely today in Convolutional Neural Networks (CNNs) for the extraction of features from images, where the kernel pixel values are learned by the network, in contrast to image processing where the kernel values are set to capture a particular feature. In this work, we use this method to extract different features of an image which are later combined as given in the paper [1] to form a hybrid image - one that creates the illusion, in the viewer, of being a different entity when viewed from different distances.

2 Theory and Intuition

2.1 Image Filtering Using Convolution

Image filtering is a process of modifying images in-order to enhance certain features/properties present in it. A linear spatial operation called *Convolution* is used in achieving the same, where the output pixel values are determined as the sum of the products the corresponding pixels of the image and the kernel(filter), with the kernel centered on the pixel whose output value is to be determined. Mathematically, the convolution is obtained as :

$$I[x,y] * K[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} I[n_1,n_2].K[x-n_1,y-n_2]$$

The output pixel value is thus obtained as a weighted sum of its neighbourhood pixels, where the weights are determined by the kernel pixel values, for e.g., a kernel whose values depict a *Gaussian* distribution results in a Gaussian smoothening of the input image.

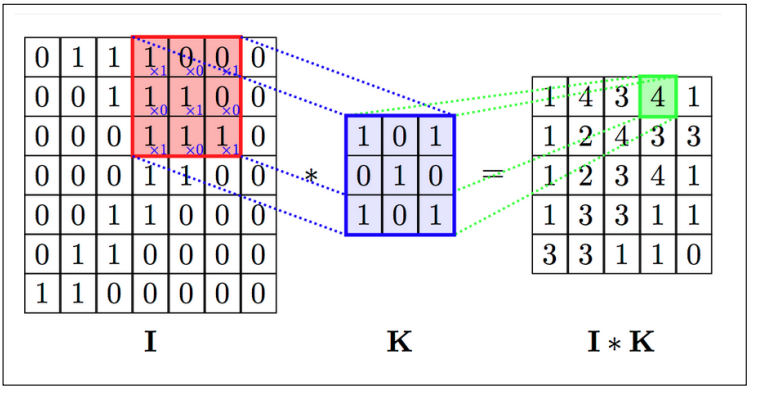

Figure 1: An illustration of a 2D convolution

2.2 Hybrid Images

The main intuition beneath the creation of hybrid images is that humans visually perceive different features of the same image at different distances. As presented in the paper by Aude Oliva[1], we focus more on high frequency components of an image when viewed from a short distance whereas the low frequency components of an image attracts visual perception at longer distances. This is the underlying principle utilized in the synthesis of the hybrid images here.
Given two spatially similar images of two different entities, we use the image filtering technique discussed above to obtain the low frequency component of one image and the high frequency component of the second image, with the help of a low pass filter and a high pass filter. The low pass filter we use in this work is a Gaussian filter. The high frequency component is obtained by subtracting the low frequency component of an image from itself.

The hybrid image H is obtained from the images I_1 and I_2 , using a Gaussian filter G as:

$$H = I_1.G + I_2.(1 - G)$$

3 Implementation

The implementation is in python and consists of the following 3 major modules :

my_imfilter.py - that takes as input any arbitrary sized image and a filter of odd size and returns the filtered image as output, by performing a convolution using first principles. The module supports both *RGB* as well as *Grayscale* images. In-order to maintain the resolution of the output image, the input image is **zero padded** relative to the size of the kernel.

proj1.py - that generates hybrid images of the inputs by generating suitable Gaussian filter for each pair of images to be hybridized and saves the plot.

vis_hybrid_image.py - that accepts a hybrid image as inputs and generates several down-scaled versions of it so that the change in interpretation is observable.

Low Pass Image	High Pass Image	Cut off Frequency Used	Kernel Size
Cat	Dog	12	49 x 49
Dog	Cat	7	29 x 29
Bird	Plane	4	17 x 17
Plane	Bird	6	25 x 25
Bicycle	Motorcycle	7	29 x 29
Motorcycle	Bicycle	6	25 x 25
Einstein	Marilyn	3	13 x 13
Marilyn	Einstein	4	17 x 17
Fish	Submarine	5	21 x 21
Submarine	Fish	3	13 x 13

Table 1: Experimentally determined cutoff frequencies used to generate the gaussian filters

4 Results and Observations

The hybrid images were generated for each compatible pair of given input images, and all different configurations were experimented, which included testing several cut off frequency for the filter used and by interchanging the low pass and high pass images. Cut of frequencies for decent hybridized images which could be interpreted differently from different distances were tabulated and chosen as the optimum cut off frequency. The optimum frequencies are tabulated in Table 1.
The kernel size was set to $(4 * cut\_off\_frequency + 1)$ x $(4 * cut\_off\_frequency + 1)$. Thus varying the cut off frequency also resulted in varying kernel size thus causing variations in the amount of blurring in the low pass output and corresponding inverse variation in the high pass output. As different pairs of compatible images were diverse in their features and properties, it is thus expected that different personalized cut off frequency of the filter would be needed for each pair.

It was also observed that the High Pass image outputs were becoming much noisier as the cutoff frequency was increased.
The hybrid images thus generated could be perceived to be different entities when scaled down (equivalent to viewing from distance). The results are given in figures 2-11 and are easily verifiable.

References

[1] Aude Oliva, Antonio Torralba, and Philippe G Schyns. Hybrid images. *ACM Transactions on Graphics (TOG)*, 25(3):527–532, 2006.

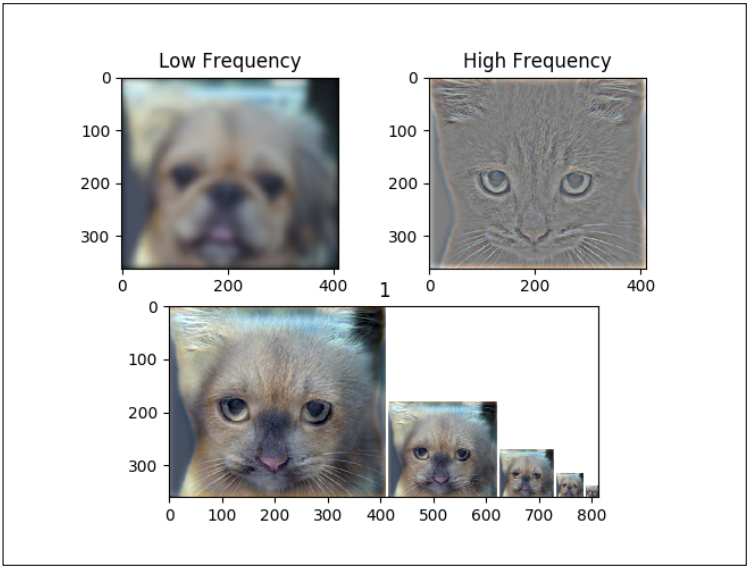

Figure 2: Hybridizing Dog(Low Pass) and Cat(High Pass)

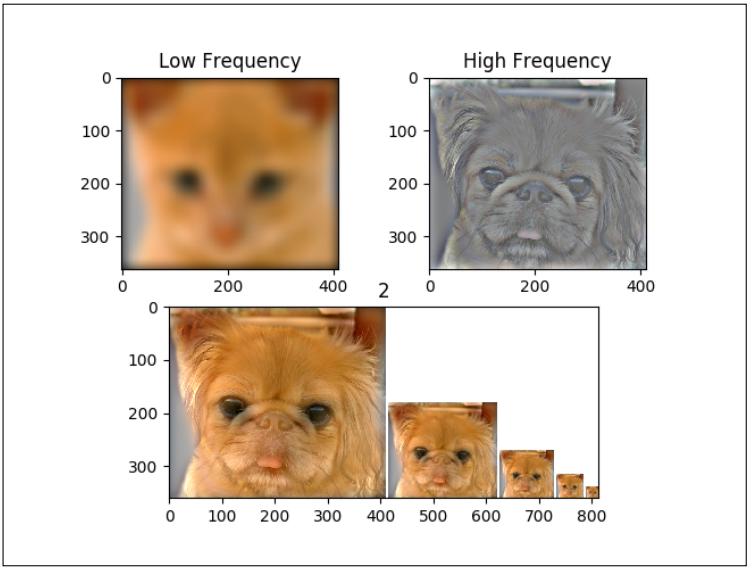

Figure 3: Hybridizing Cat(Low Pass) and Dog(High Pass)

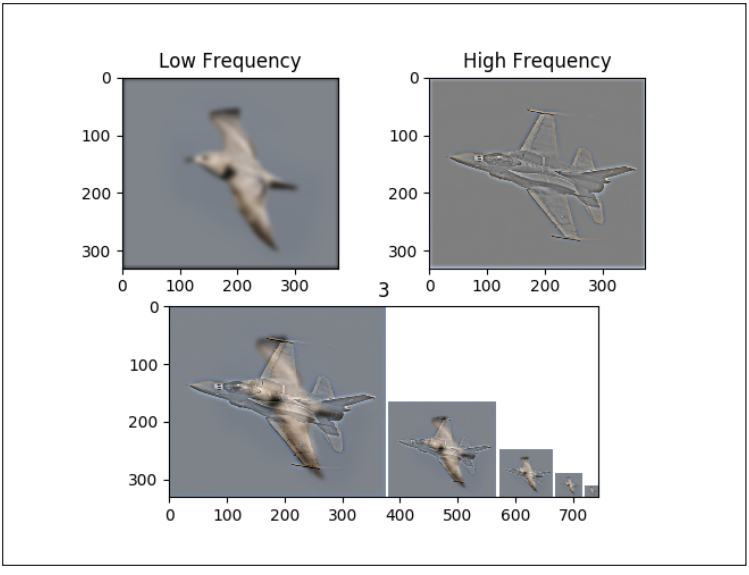

Figure 4: Hybridizing Bird(Low Pass) and Plane(High Pass)

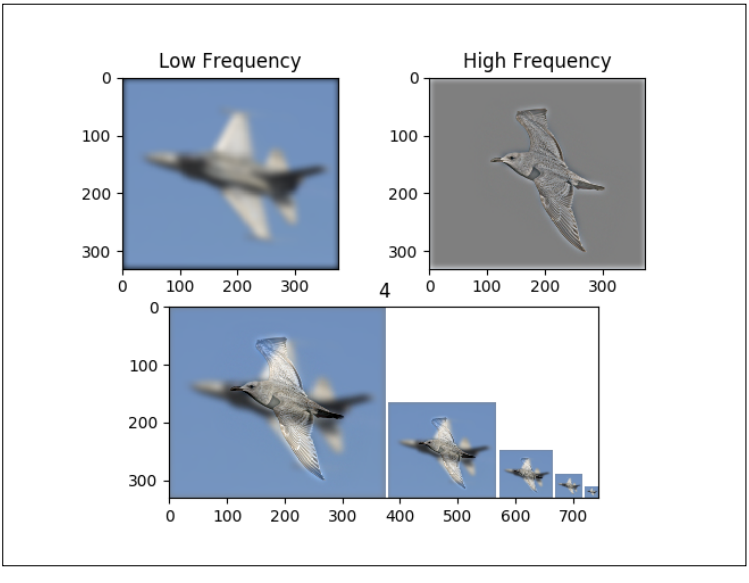

Figure 5: Hybridizing Plane(Low Pass) and Bird(High Pass)

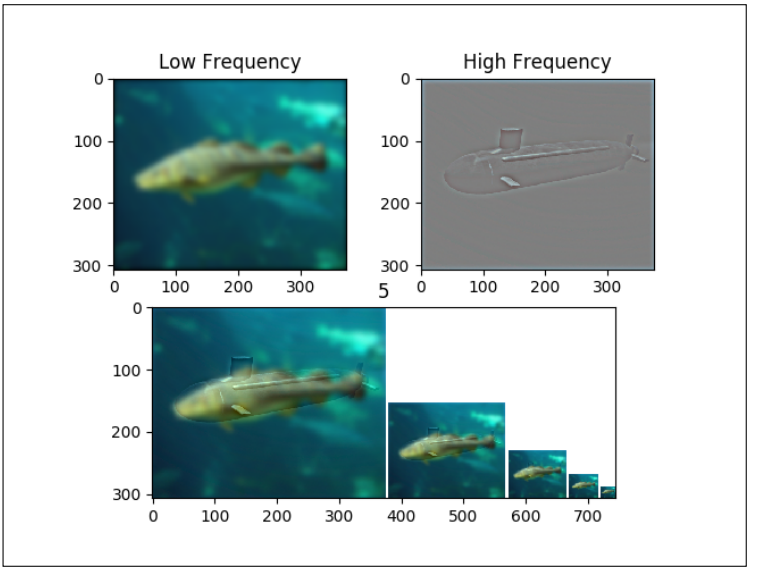

Figure 6: Hybridizing Fish(Low Pass) and Submarine(High Pass)

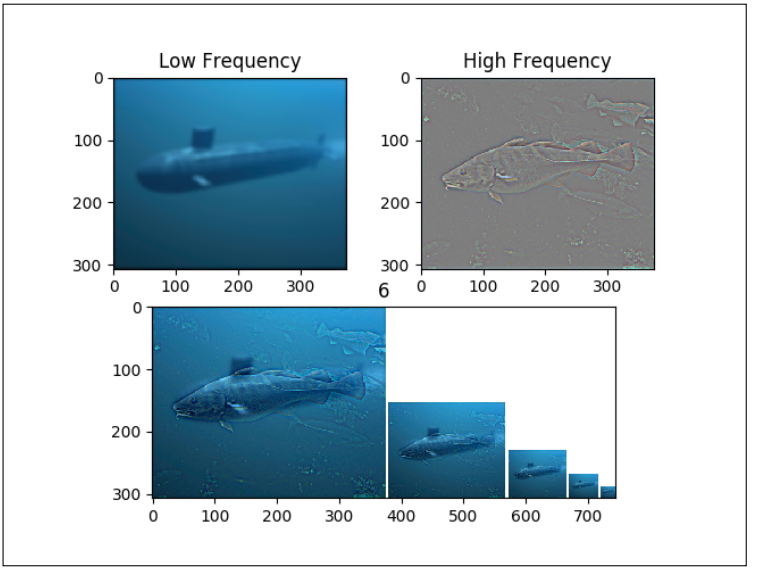

Figure 7: Hybridizing Submarine(Low Pass) and Fish(High Pass)

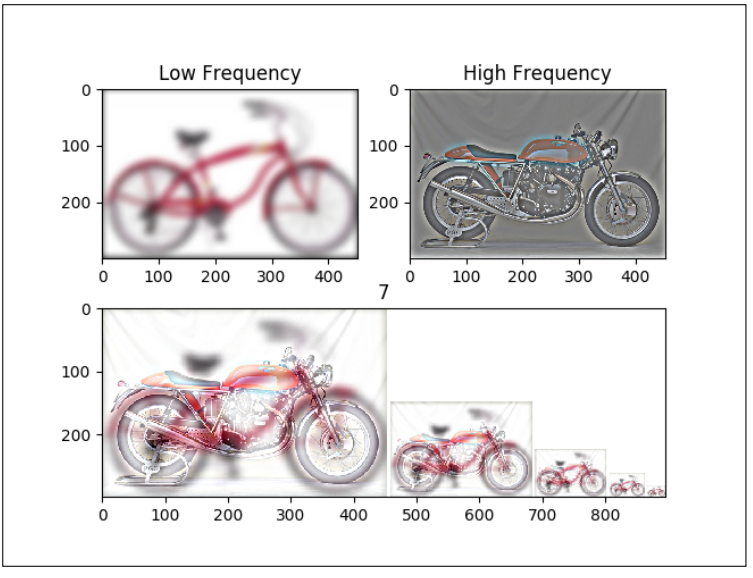

Figure 8: Hybridizing Bicycle(Low Pass) and Motorcycle(High Pass)

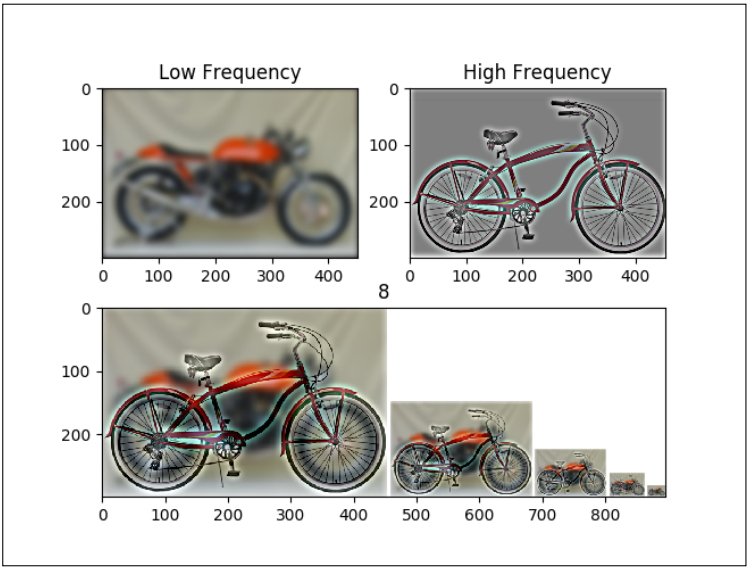

Figure 9: Hybridizing Motorcycle(Low Pass) and Bicycle(High Pass)

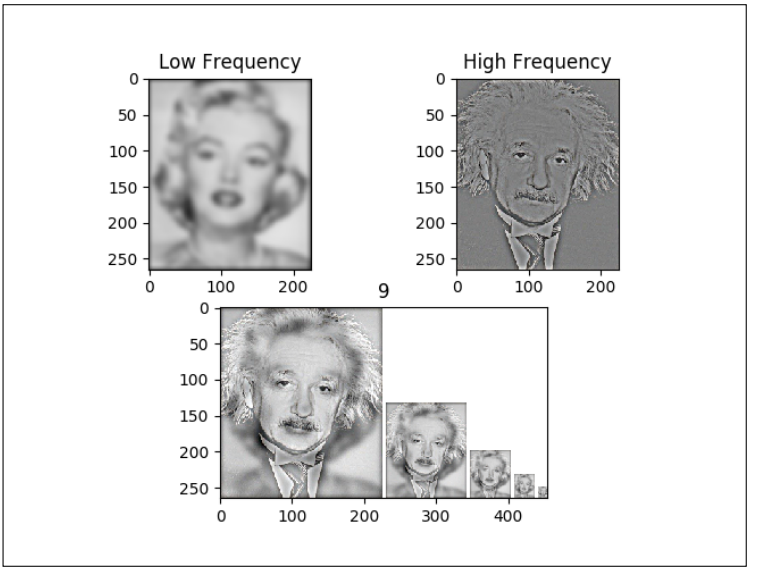

Figure 10: Hybridizing Marilyn(Low Pass) and Einstein(High Pass)

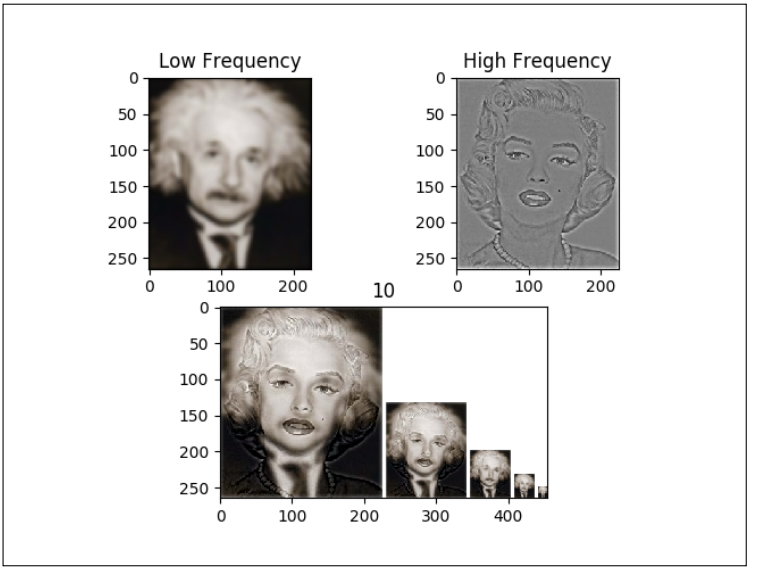

Figure 11: Hybridizing Einstein(Low Pass) and Marilyn(High Pass)