Image filtering and Hybrid Images

Vishal Yadav
2018csb1128
2018csb1128@iitrpr.ac.in

Indian Institute of Technology
Ropar, 140001
Punjab, India

Abstract

Hybrid image is the combination of a low frequency image and a high frequency image. First introduced by Oliva, Torralba, and Schyns in the SIGGRAPH [1] 2006 paper. A charactor of the hybrid image is that the image looks different from different distance. This is because the one can only see low frequency part of the image from distance. In this assignment, I will implement a simplified version of the Hybrid Image.

1 Introduction

Image filtering (or convolution) is a fundamental image processing tool. More specifically, you will implement *my_imfilter()*, an image filtering function to create Hybrid Images without using any built-in efficient image filtering function of Python.
A **Hybrid image** is the sum of a low-pass filtered version of the one image and a high-pass filtered version of a second image. There is a free parameter, which can be tuned for each image pair, which controls how much high frequency to remove from the first image and how much low frequency to leave in the second image. This is called the "cutoff-frequency". In the paper it is suggested to use two cutoff frequencies (one tuned for each image) and you are free to try that, as well. In the starter code, the cutoff frequency is controlled by changing the standard deviation of the Gausian filter used in constructing the hybrid images.

The key idea is that high frequency dominates perception when visible, but, at a distance, only the low frequency (or smooth) part of the image signal is visible. By blending the high frequency components of one image with the low-frequency components of another, you synthesize a hybrid image which induces different interpretations at different distances.

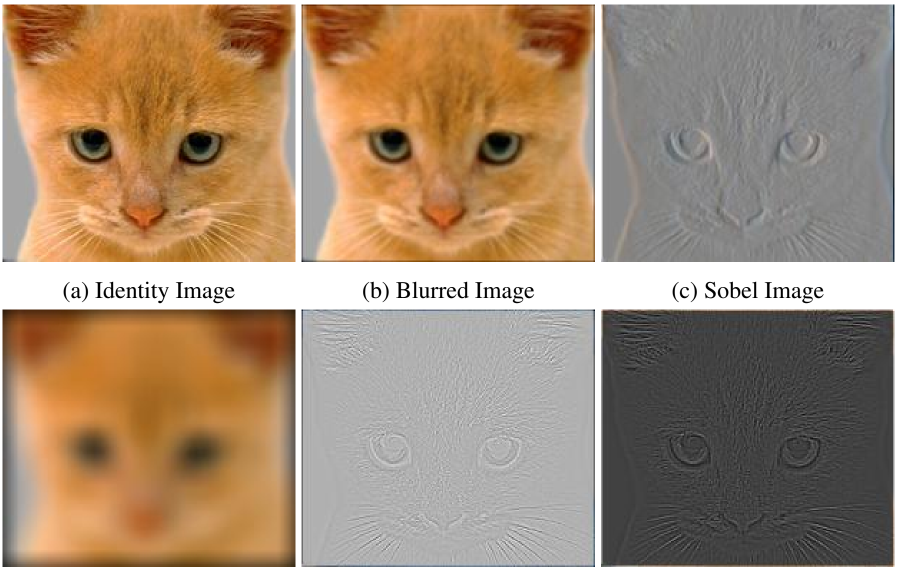
More specifically, our new function **my_imfilter()** will replicate the default behavior of the function(s) *scipy.misc.imfilter() or scipy.ndimage.filters.correlate()*.

2 Methodology

How We Filter Images: Essentially what we do implementation-wise, is treat the image and filter as two matrices. We pad the image an appropriate amount depending on the size of the filter. There are multiple possiblities in terms of padding (mirror the image, extend the image, perhaps even speculate the image, lots of options here), but for the purpose of the project we just chose to leave the padded values as 0. We then slide the filter through each part of the image, to make a new image (that has the same dimensions of the original image) as the filtered output by recording the dot product at each part of the padded image we place the filter over. In the context of the hybrid image problem, this works for creating a low frequency filter by just replacing the color at a given location with something similar to what the other colors are like. On the other hand, if this were negated (for the high frequency filter), it would essentially tone down a lot of the surrounding color and in its place sharpen the distinct edges. Both these two possibilities come into play for making the hybrid images. Matrix magic is the backbone of the image filtering process.

Creating Hybrid Images: To create a hybrid image, two images need to be superimposed at different spacial scales. One image needs to be filtered with a low-pass filter, a blurring or smoothening filter. This low-pass filter will average out rapid changes in intensity. The other image needs to be filtered with a high-pass filter, also known as shaprening. This filter will allow one to see finer details in the image. Note that a low-pass filter is the opposite of a high-pass filter. Every image taken will have some noise to it. The low-pass filter will smooth out this noise, while the high-pass filter will amplify this noise. Once we have our low-filtered image and high-filtered image, we simply add these two images together.

Mathematical: Let H represent a hybrid image, I1 represent image 1, I2 represent image 2, G1 represent a low-pass filter, and (1 G2) represent a high-pass filter. We can then use the following equation to generate our hybrid image: H = I1*G1 + I2*(1-G2)



(a) Identity Image (b) Blurred Image (c) Sobel Image

(d) Large Blur Image (e) Laplacian Image (f) High Pass Image

Figure 1: Different images obtained via proj1_test_filtering.py

3 Algorithm

The algorithm used attempts to merge the low-pass filtered version of one image and the high-pass filtered version of another to produce the hybrid image by three steps as below:
(1) Get dimensions of the new padded matrix and initialize with zeroes
(2) Align the input image in center of this new padded matrix (We will convolute/correlate our filter with this new padded matrix that ensures us output image of same shape as input image)
(3) Correlate the padded matrix with our filter for each index in the final output image and setting the value of the pixel to the sum of matrix(convoluted with filter).

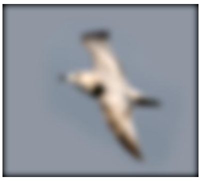
Algorithm 1 my_imfilter(img, imfilter)

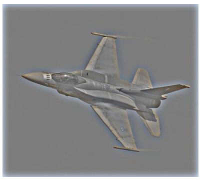
```
 1: $output \leftarrow img.copy()$
 2: $pad\_length \leftarrow (imfilter\_length - 1)/2$
 3: $pad\_width \leftarrow (imfilter\_width - 1)/2$
 4: $final\_shape \leftarrow (img\_length + 2 * pad\_length, img\_width + 2 * pad\_width, num\_ch)$
 5: $padded\_matrix \leftarrow matrix\_of\_zeroes(shape = final\_shape)$
 6: $padded\_matrix[pad\_length : img\_length + pad\_length, pad\_width : img\_width + pad\_width] \leftarrow img$
 7: for $k \leftarrow 0 : img\_channels$ do
 8:     for $i \leftarrow 0 : img\_length$ do
 9:         for $j \leftarrow 0 : img\_width$ do
10:             $mult\_mat \leftarrow mat\_mult(padded\_matrix[i : i + fil\_length, j : j + fil\_width, k], imfilter)$
11:             $output[i, j, k] \leftarrow sum\_matrix(mult\_mat)$
12:         end for
13:     end for
14: end for
15: return output
```

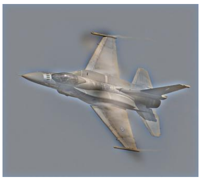
4 Results and Inferences

I created hybrid image from five image pairs given. The result of them is satisfying. But there is a need to change the cutoff frequency for each pair of images. Because some low frequency image has more edges, so I need to blur it more so that the image will not be seen in the hybrid image from a short distance. Different choice of which image to be low frequency image will also effect the outcome. In general, the main object in low frequency should not be smaller than the object in high frequency. Otherwise it will make the low frequency image hard to distinguish.

Also, if we use color for both the images to be merged, the low-pass filtered version of the image holds more color information than the high-pass. Therefore, to use color for the low-frequency component will often significantly reduce the perception of the high-frequency component of the other image. However, to use color only for the high-frequency component will produce a better hybrid result.


(a) Low Freq


(b) High Freq


(c) Hybrid Image

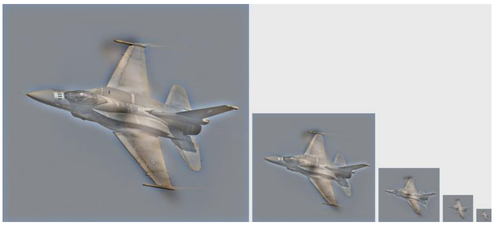
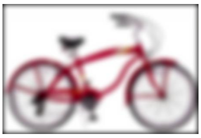
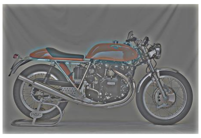
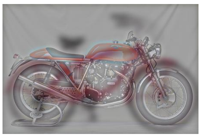

(d) Scales

Figure 2: Different images obtained via proj1.py using a pair of images(Bird-Plane)


(a) Low Freq


(b) High Freq


(c) Hybrid Image

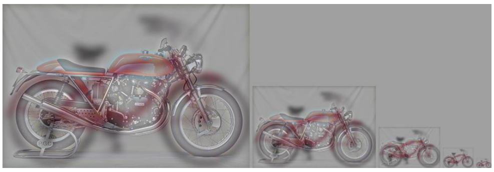
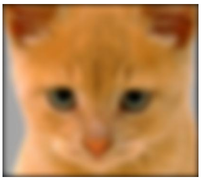

(d) Scales

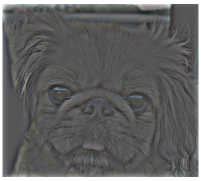
Figure 3: Different images obtained via proj1.py using a pair of images(Bicycle-Motorcycle)

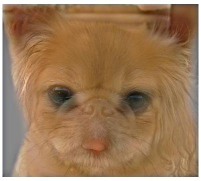
One interesting application listed in the SIGGRAPH 2006 paper by Oliva, Torralba, and Schyns is the private font to use hybrid images to display text that is not visible for people standing at some distance from the screen.
The results obtained from the code are shown:

References

[1] Philippe. G. Schyns Aude Oliva, Antonio Torralba. Hybrid images. 2006.


(a) Low Freq


(b) High Freq


(c) Hybrid Image

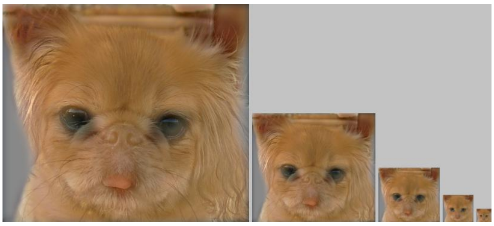
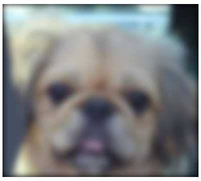
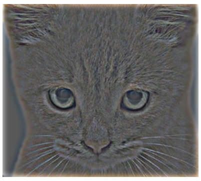
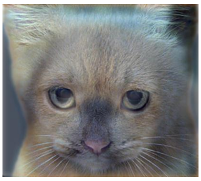

(d) Scales

Figure 4: Different images obtained via proj1.py using a pair of images(Cat-Dog)


(a) Low Freq


(b) High Freq


(c) Hybrid Image

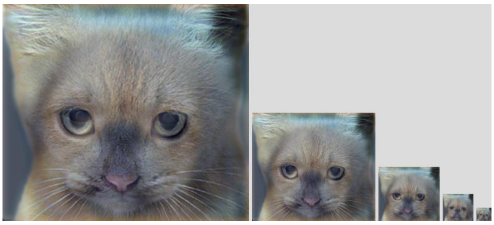
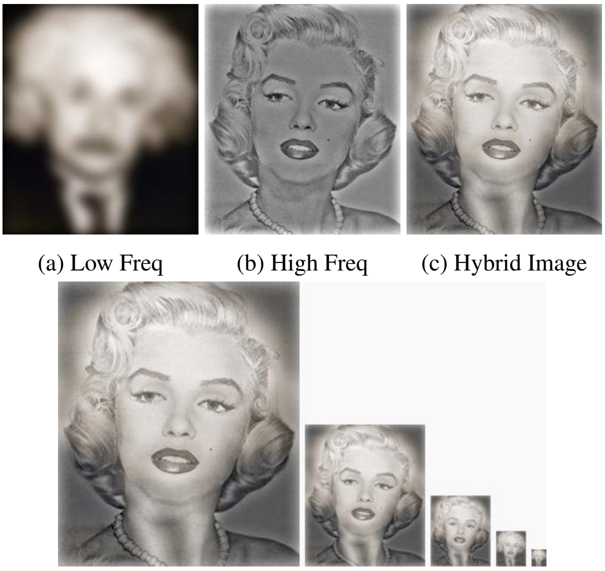

(d) Scales

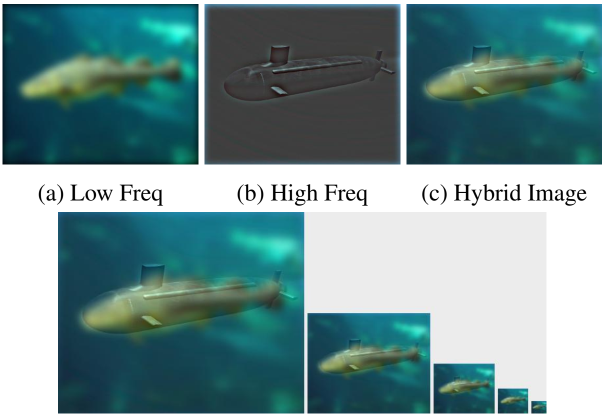
Figure 5: Different images obtained via proj1.py using a pair of images(Dog-Cat)



(a) Low Freq (b) High Freq (c) Hybrid Image

(d) Scales

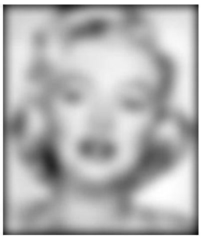
Figure 6: Different images obtained via proj1.py using a pair of images(Einstein-Marilyn)

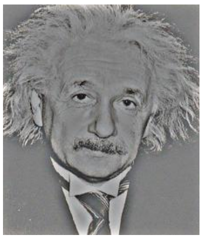


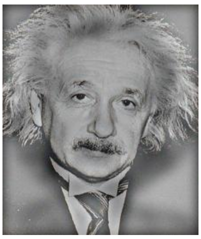
(a) Low Freq (b) High Freq (c) Hybrid Image

(d) Scales

Figure 7: Different images obtained via proj1.py using a pair of images(Fish-Sub)


(a) Low Freq


(b) High Freq


(c) Hybrid Image

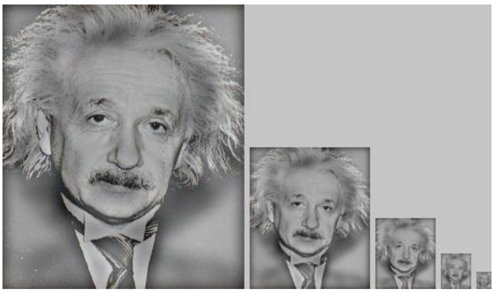
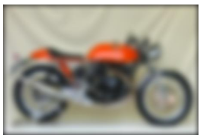
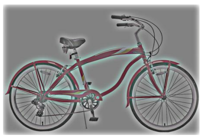
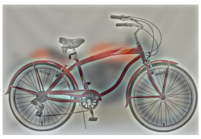

(d) Scales

Figure 8: Different images obtained via proj1.py using a pair of images(Marilyn-Einstein)


(a) Low Freq


(b) High Freq


(c) Hybrid Image

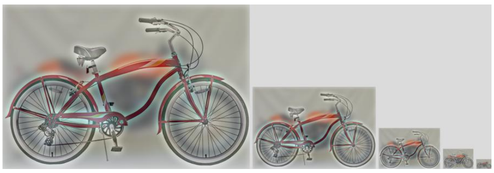
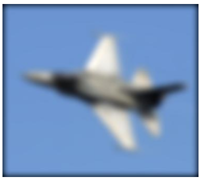
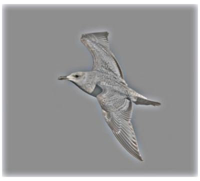
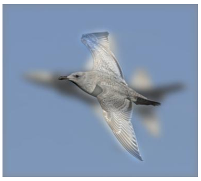

(d) Scales

Figure 9: Different images obtained via proj1.py using a pair of images(Motorcycle-Bicycle)


(a) Low Freq


(b) High Freq


(c) Hybrid Image

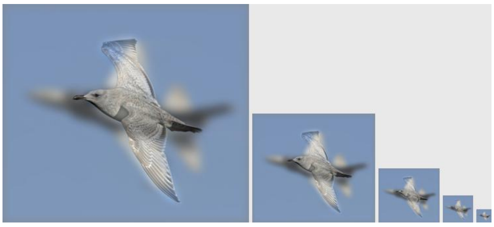
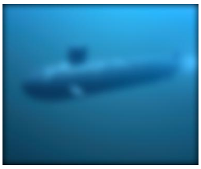
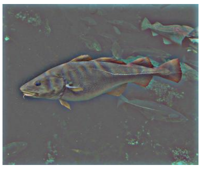
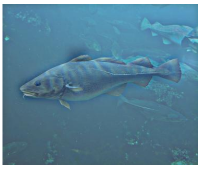

(d) Scales

Figure 10: Different images obtained via proj1.py using a pair of images(Plane-Bird)


(a) Low Freq


(b) High Freq


(c) Hybrid Image

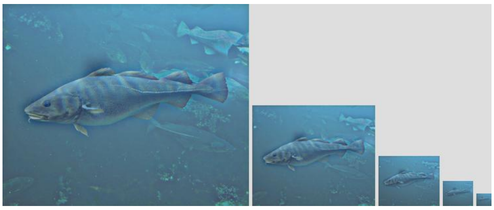
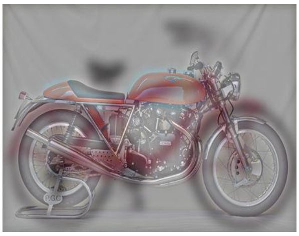
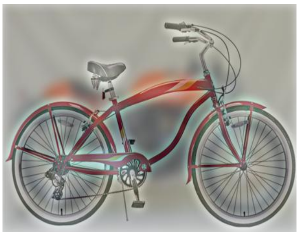
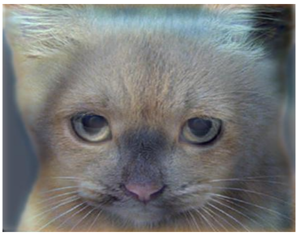
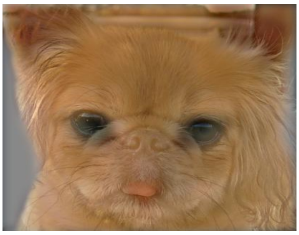

(d) Scales

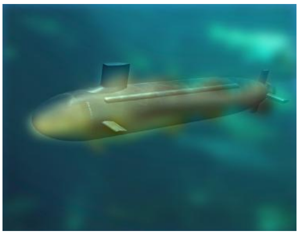
Figure 11: Different images obtained via proj1.py using a pair of images(Sub-Fish)

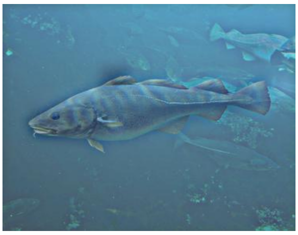

(a) Bicycle-Motorcycle

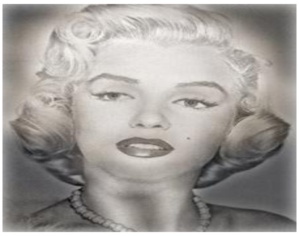

(b) Motorcycle-Bicycle

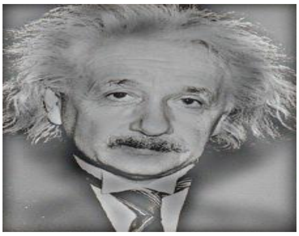

(c) Dog-Cat


(d) Cat-Dog


(e) Fish-Submarine


(f) Submarine-Fish


(g) Einstein-Marilyn


(h) Marilyn-Einstein

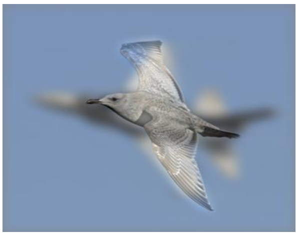

(i) Plane-Bird

Figure 12: Different images obtained via proj1.py using a pair of images