Hybrid Images and Filters

Aman Bilaiya
2018csb1069

Indian Institute Of Technology Ropar
Rupnagar, Punjab

Abstract

Today, with advance libraries languages provide various in-built image filtering functions. This report shows the implementation of my own image filtering function (may not be as efficient as in-built functions) using matrix computations which works similar to imfilter() of Matlab and tested upon various filters like identical, low pass, high pass, sobel etc.
Used this filtering function to synthesise hybrid images using a simplified version of the SIGGRAPH 2006 paper by Oliva, Torralba, and Schyns [1].

1 Introduction

1.1 Image Filters

Image filtering or convolution is a fundamental image processing technique. It helps in modifying or enhancing an image. This technique involves sliding a kernel over input image and performing convolution sum to update pixel value at the centre of the kernel. Different types of kernel can be used to perform different tasks like smoothing, sharpening, and edge detection etc.

In this paper, we describe our algorithm to implement own image filtering function which can be used for both grey scales and color images. To handle border pixels we use padding with zero, which is adding zero pixels around the image to ensure proper positioning of kernel/filter on the input image around the corners.

1.2 Hybrid Images

Hybrid images are static images that changes in interpretation as a function of the viewing distance. At a distance, only the low frequency part of the image signal is visible while with close look we are able to see higher frequency components.

Hybrid images is the combination of low pass component of one image and high pass component of other generated by our filter function, we visualise the hybrid image by progressively down-sampling the images and concatenating them together.

2 Implementation

2.1 Filtering Method

In order to implement my_imfilter(), we will take two inputs: 1) input image 2) filter. We will use a padded matrix with center adjusted to the input image and boundary padded with zeroes as per the filter dimensions. Then we will convolve filter over this padded image and generate the filtered image reshaped to its original size.

Algorithm 1 my_imfilter (img, fil)

channels $\leftarrow len(img[0][0])$
output $\leftarrow zeros\_mat(img.H, img.W, channels))$
padded_img $\leftarrow zeros\_mat(img.H + fil.H - 1, fil.W + fil.W - 1, channels)$

padded_img$[int((fil.H - 1)/2) : img.H + int((fil.H - 1)/2), int((fil.W - 1)/2) : img.W + int((fil.W - 1)/2)] \leftarrow$ **img**

for $k \leftarrow 0 : channels$ **do**
  for $i \leftarrow 0 : img.H$ **do**
    for $j \leftarrow 0 : img.W$ **do**
      $output[i][j][k] \leftarrow np.sum(np.multiply(padded\_img[i : i + fil.H, j : j + fil.W,\ k], fil))$
    end for
  end for
end for
$output \leftarrow clip(output, 0, 1)$
return output

NOTE: Here H is the height and W is the width of the corresponding object. Also img is the input image & fil is the Filter.

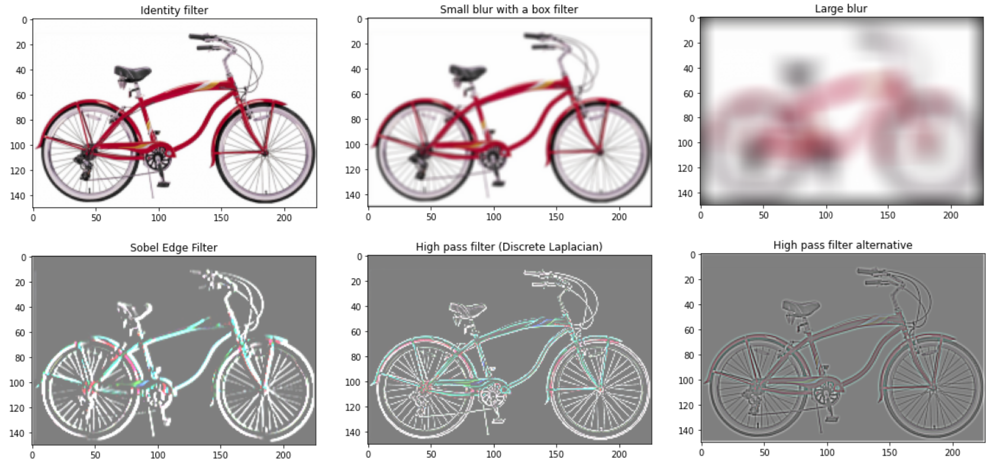

Figure 1: Filtered Images obtained from proj1_test_filtering.py

2.2 Hybrid Image Synthesis

Algorithm 2 Hybrid Image

filter $\leftarrow fspecial(Gaussian, cutoff\_frequency * 4 + 1, cutoff\_frequency)$
low_freq $\leftarrow my\_imfilter(image1, filter)$
high_freq $\leftarrow image2 - my\_imfilter(image2, filter)$
hybrid_image $\leftarrow low\_freq + high\_freq$
hybrid_image_scales $\leftarrow vis\_hybrid\_image(hybrid\_image)$
return hybrid_image

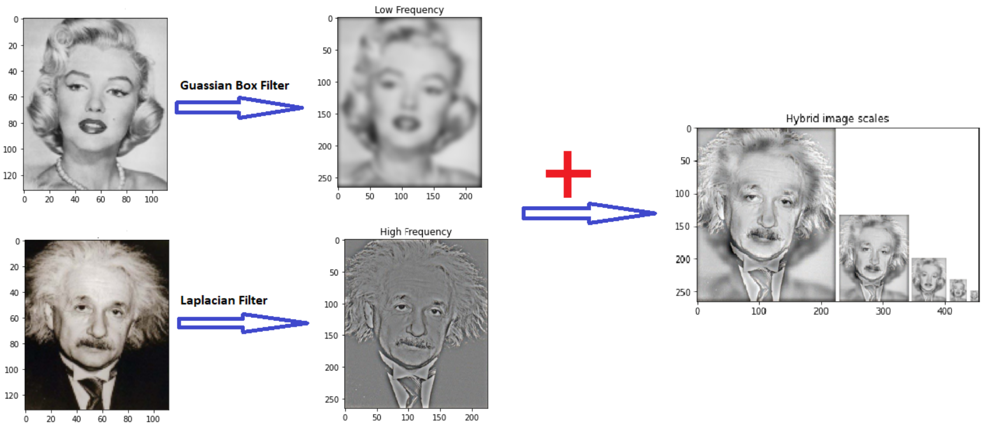

Figure 2: Ilustration of Hybrid Image Synthesis

3 Results & Observations

1) Generated hybrid images for each properly aligned input image pairs and tried different configurations including switching low pass and high pass images and changing cutoff frequency for the filter.

2) Adjusting the cutoff frequency also resulted in varying filter size [4f+1,4f+1] thus causing variations in the extent of blurring in the low pass component and corresponding sharpening in the high pass component. For each pair of images, a particular cutoff frequency between [2-12] gave the best hybrid image output.

3) High Pass image component was observed to be much noisier as the cutoff frequency was increased.

4) We can see from the hybrid image scales in figures 3-12 that initially the high frequency component dominates but as the scale reduces, low frequency components dominates and catches our eye. Thus hybrid images is perceived differently when scaled down.

Below are the best hybrid images obtained by varying cutoff frequency :-

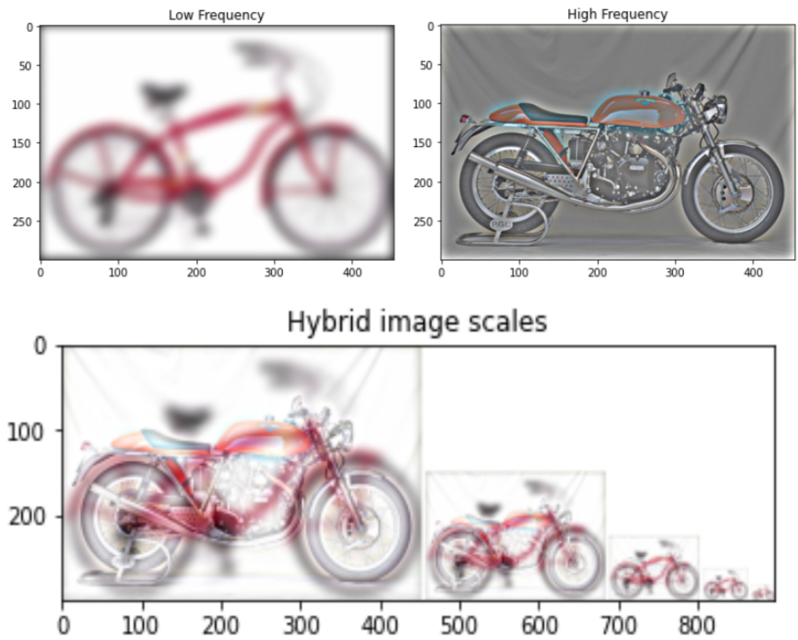

Fig 3: Cycle (Low) and Bike (High) Hybrid

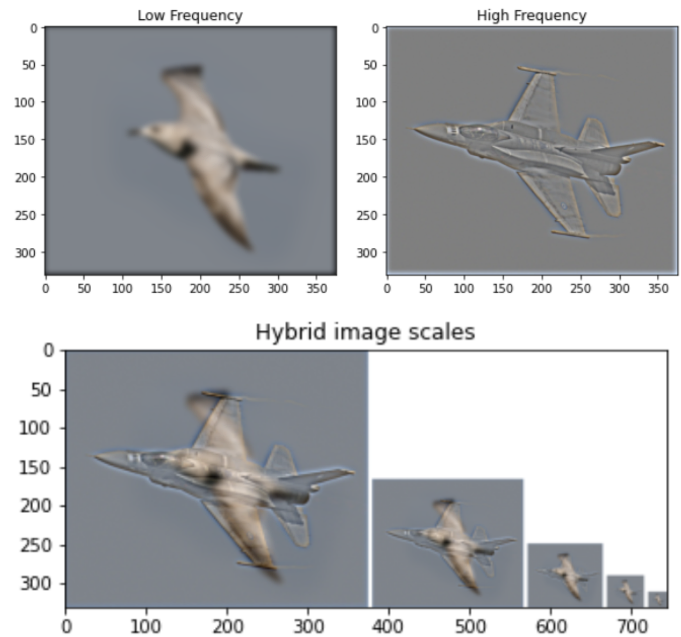

Fig 4: Bird (Low) and Plane (High) Hybrid

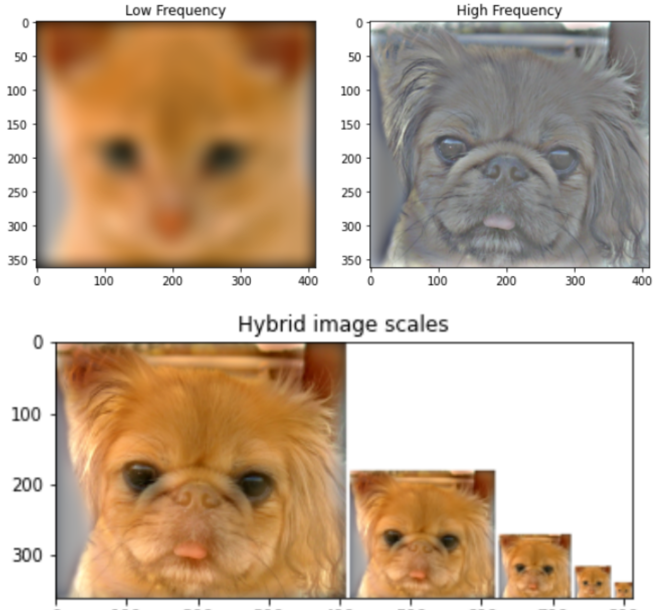

Fig 5: Cat (Low) and Dog (High) Hybrid

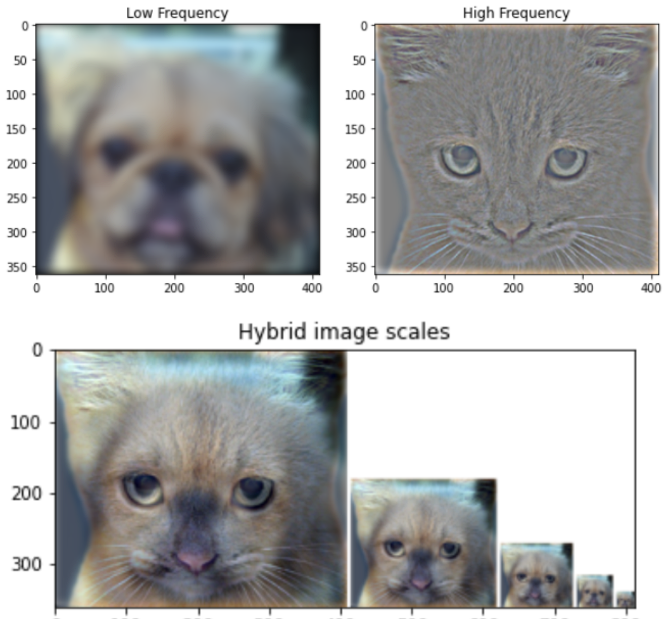

Fig 6: Dog (Low) and Cat(High) Hybrid

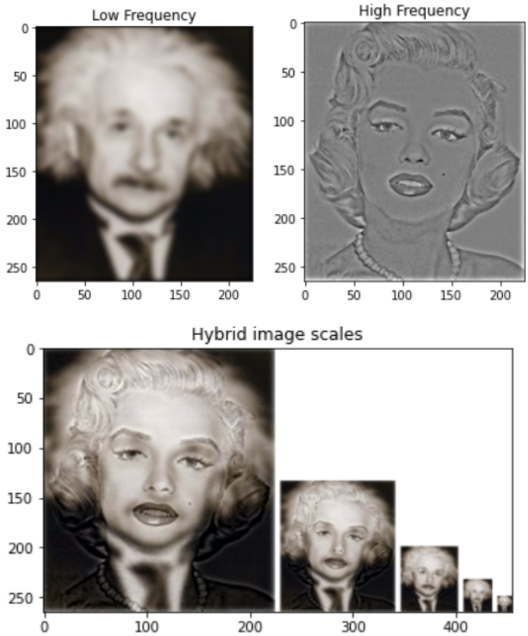

Fig 7: Einstein (Low) and Marilyn(High) Hybrid

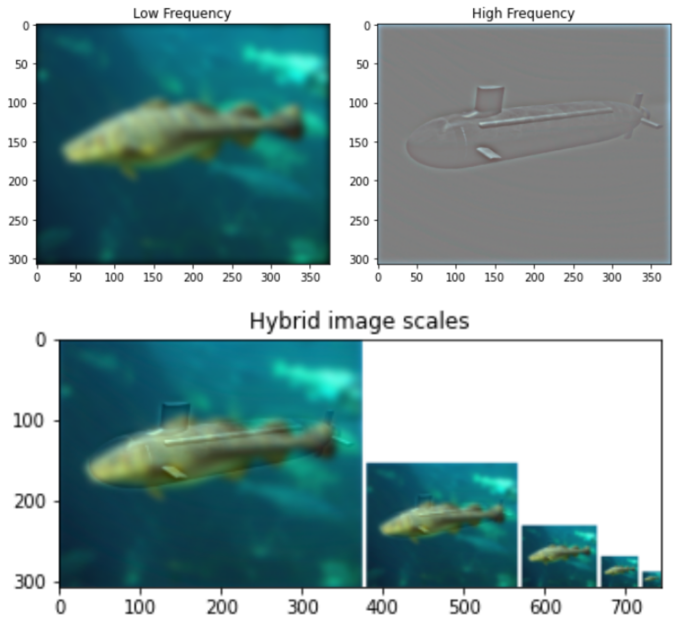

Fig 8: Fish (Low) and Submarine (High) Hybrid

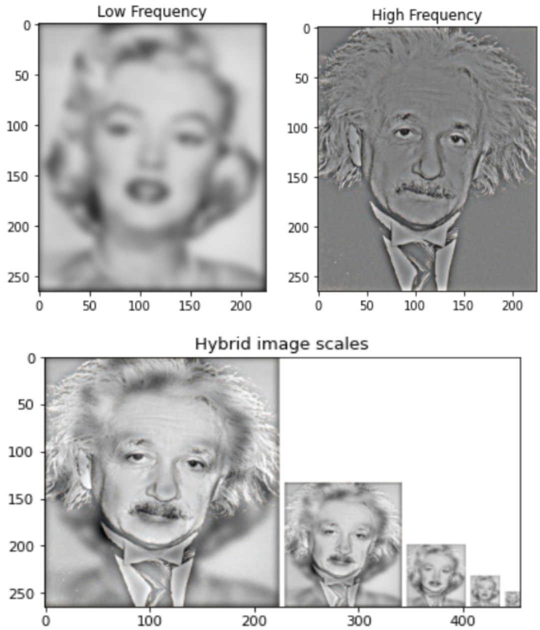

Fig 9: Marilyn (Low) and Einstein (High) Hybrid

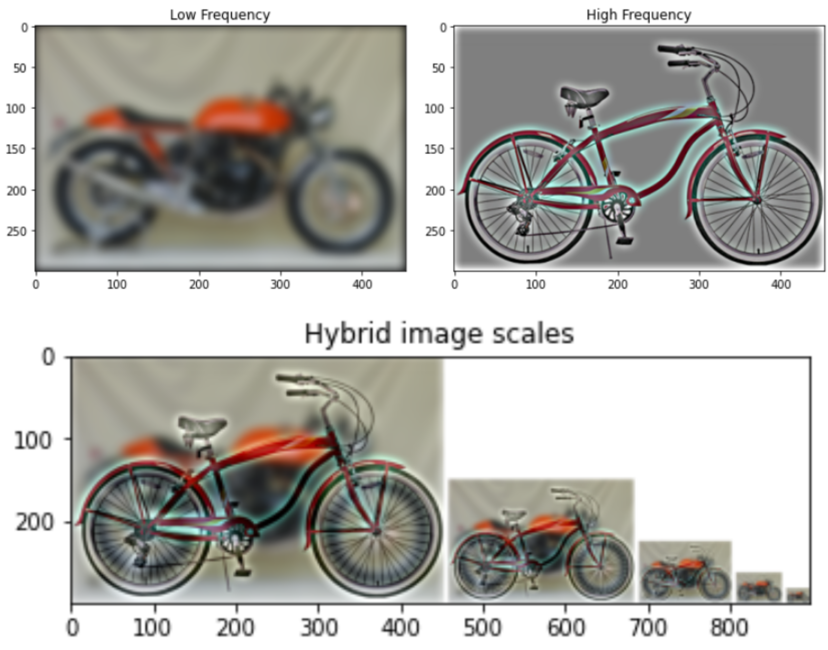

Fig 10: Bike (Low) and Cycle (High) Hybrid

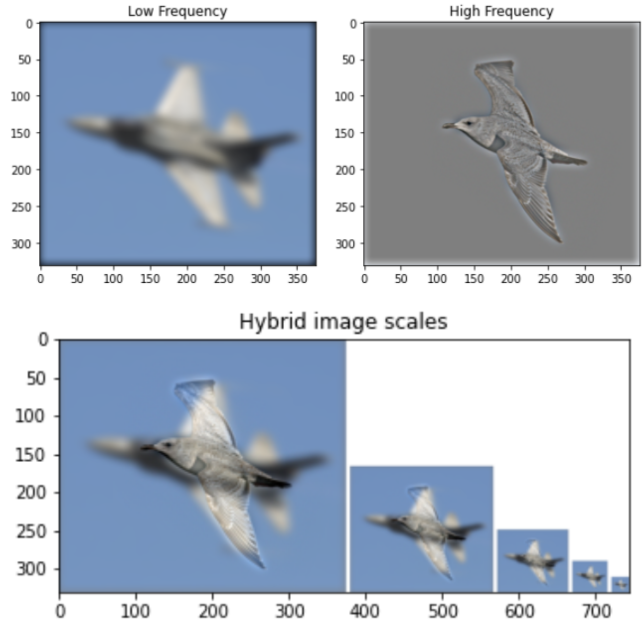

Fig 11: Plane (Low) and Bird (High) Hybrid

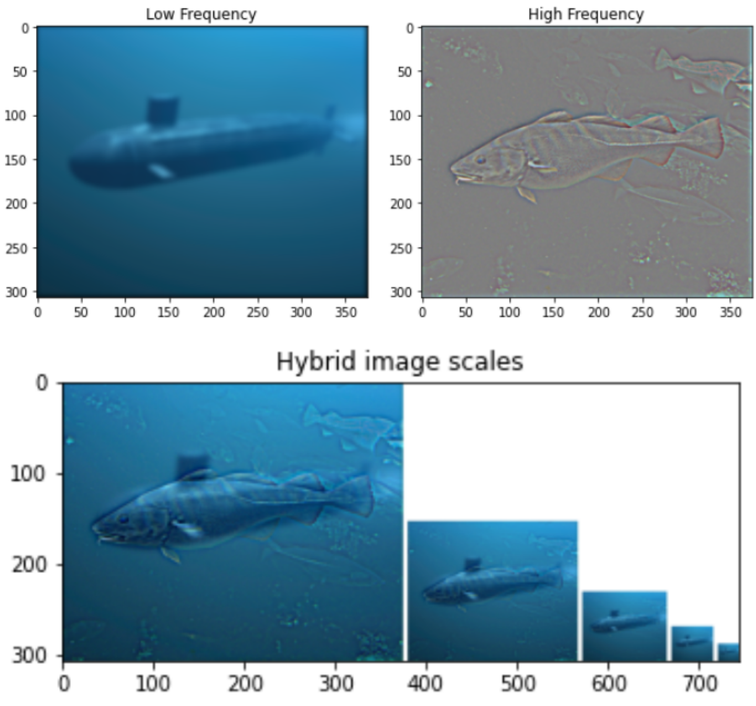

Fig 12: Submarine (Low) and Fish (High) Hybrid

Low Freq Image	High Freq Image	Cutoff Freq	Kernel Size [4f+1,4f+1]
Bicycle	Motorcycle	7	29 x 29
Bird	Plane	4	17 x 17
Cat	Dog	11	45 x 45
Dog	Cat	7	29 x 29
Einstein	Marilyn	3	13 x 13
Fish	Submarine	5	21 x 21
Marilyn	Einstein	4	17 x 17
Motorcycle	Bicycle	6	25 x 25
Plane	Bird	6	25 x 25
Submarine	Fish	3	13 x 13

Note :- Table shows the cutoff frequency used for given input image pairs which gave best hybrid output image

References

[1] SIGGRAPH 2006 paper by Oliva, Torralba, and Schyns
[2] Online Resources