Filters and Hybrid Images

Harshit Malik
2017csb1078@iitrpr.ac.in

Indian Institute of Technolgy
Ropar, Punjab
India

Abstract

Today, with advance liberaries, programming languages has lot to offer for programmer as pre-defined functions. Applying filters on images is one such example. Matlab has imfilter() method python has scipy.ndimage.correlate(). This paper is in the direction of implementing your own filtering method with basic matrix operation.

In this paper we have used our filtering method to produce **Hybrid Images**, using the methodology of simplified version of the *SIGGRAPH 2006 paper by Oliva, Torralba, and Schyns*.

1 Introduction

1.1 Image Filters

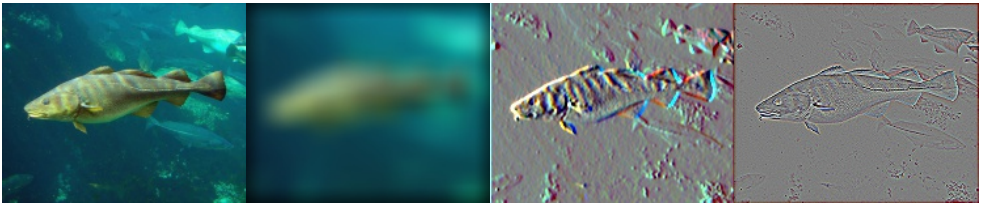
Image filtering (or convolution) is a fundamental image processing tool. It is a technique for modifying or enhancing an image. This technique involves sliding a kernel over input image and performing convolution sum to update pixel value at the centre of the kernel. Different types of kernel can be used to perform defferent tasks like smoothing, sharpening, and edge enhancemen etc.
In this paper, we propose methodology to implement your image filtering method 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 postioning of kernel on the input image around the corners.

1.2 Hybrid Images

A hybrid image is an image that is perceived in one of two different ways, depending on viewing distance, based on the way humans process visual input. Hybrid images combine the low spatial frequencies of one picture with the high spatial frequencies of another picture, producing an image with an interpretation that changes with viewing distance.

2 Methodology

To implement im_filter method to mimic imfilter() (Matlab) and scipy.ndimage.correlate() (Python), we will take two inputs: 1) input image 2) filter.We will use a temporary padded matrix with centre as input image and boundry padded with zero pixels as per size of filter. Then we will convolve over this padded image to generate output image.



(a) Identity (b) Blurred (c) Sobel (d) Laplacian

Figure 1: Output of proj_test_filtering

Algorithm 1 my_imfilter(img, imfilter)

$padded\_img \leftarrow zero\_matrix(img.length + imfilter.length - 1, img.width + imfilter.width - 1, img.channels)$
$padded\_img[imfilter.length - 1)/2 : img.length + (imfilter.length - 1)/2, (imfilter.width - 1)/2 : img.width + (imfilter.width - 1)/2] \leftarrow img$
for $k \leftarrow 0 : img.channels$ **do**
  for $i \leftarrow 0 : img.length$ **do**
    for $j \leftarrow 0 : img.width$ **do**
      $temp\_mat \leftarrow matrix\_multiplication(padded\_img[i : i + imfilter.length, j : j + imfilter.width, k], imfilter)$
      $output\_img[i, j, k] \leftarrow matrix\_sum(temp\_mat)$
    end for
  end for
end for
return output_img

3 Results

We see from the figure 1 that we can perform operations like sharpening, smoothening,edge detection etc. using image filtering techniques.
We also see from the below hybrid images that the high frequencies dominates when the distance between the image and the viewer is small but as the distance increases the lower frequencies become more and more prominent.
However the hybrid images donâĂŹt work so well if the pictures are not aligned or donâĂŹt have similar shapes to begin with.


(a) Hybrid


(b) Low Frequency


(c) High Frequency


(d) Scaled images

Figure 2: Motorcycle-Bicycle

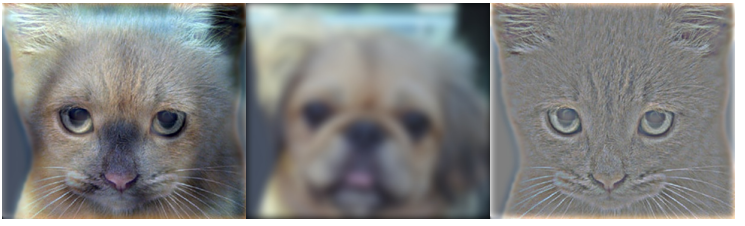

(a) Hybrid


(b) Low Frequency


(c) High Frequency


(d) Scaled images

Figure 3: Bird-Plane



(a) Hybrid (b) Low Frequency (c) High Frequency

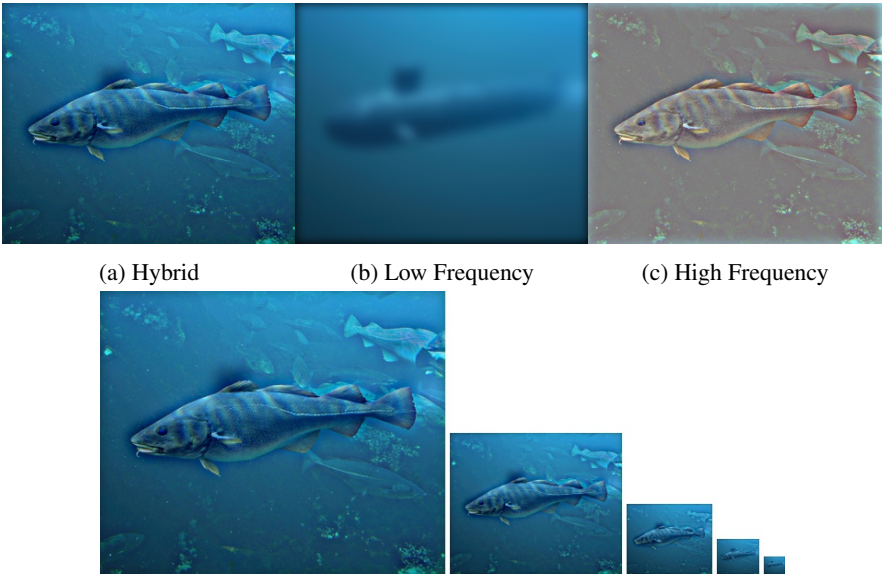
(d) Scaled images

Figure 4: Dog-Cat

(a) Hybrid (b) Low Frequency (c) High Frequency

(d) Scaled images

Figure 5: Einstein-Marilyn



(a) Hybrid (b) Low Frequency (c) High Frequency

(d) Scaled images

Figure 6: Submarine-Fish