Computer Vision : Project 1

Image Filtering and Hybrid Images

Srijan Mallick
Alimpan Barik
Semester 2
M.Sc Big Data Analytics, RKMVERI

March 2021

Contents

1 Introduction:

Hybrid images are images which have different interpretations based on the viewing distance. Up close, higher frequencies tend to dominate our vision, while from farther away we tend to only interpret the lower frequencies. By taking advantage of this interesting property of our vision, we can create ***hybrid images*** that are composed of high frequencies of one image and low frequencies of another. The technique for creating hybrid images was first developed by Aude Oliva of MIT and Philippe G. Schyns of University of Glasgow, a method originally proposed by Schyns and Oliva in 1994.

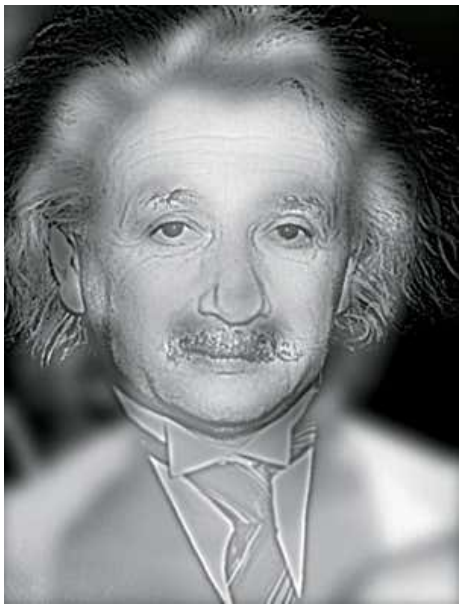


Figure 1.1: Example of a hybrid image combining
Albert Einstein and Marilyn Monroe

2 Algorithm:

Hybrid images are generated by superimposing two images at two different spatial scales: the low-spatial scale is obtained by filtering one image with a low-pass filter; the high spatial scale is obtained by filtering a second image with a high-pass filter. The final image is composed by adding these two filtered images.

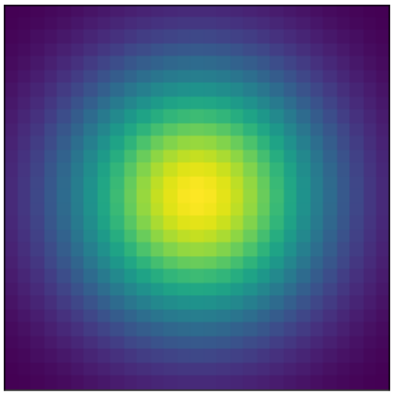
Let H represent a hybrid image, I_1 represent image 1, I_2 represent image 2 and G represent a low-pass filter. We can then use the following equation to generate our hybrid image:

$$H = (I_1 * G) + (I_2 - (I_2 * G))$$

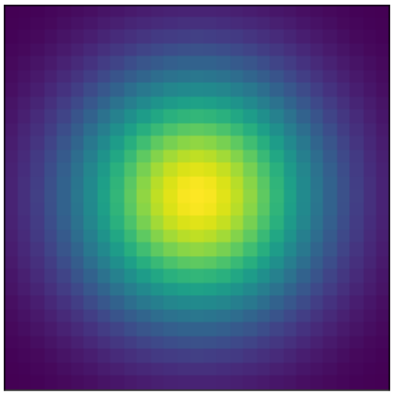
Given two images, the following steps can be used to create a hybrid image:

2.1 Creating the Gaussian filter

The Gaussian filter is a 2-D convolution operator that is used to 'blur' images and remove noise. Its impulse response looks like a Gaussian distribution, and hence the name. It is also used for sharpening, embossing, edge detection, and more. In this project, we have used OpenCV's *cv2.getGaussianKernel()* function which returns a vector, which after multiplying with its transpose, returns our desired filter. However, we can completely remove the dependency on OpenCV library by manually creating a Gaussian kernel of our own, which we see from the following image, produces the same result as the kernel derived from the *cv2.getGaussianKernel()* function.



(a) OpenCV's filter



(b) Custom made filter

2.2 Correlation:

Correlation is the process of moving a filter mask/kernel over the image, computing the weighted sum at each pixel and storing the value in a new duplicate matrix. If F be the filter and I be the image, then the 2D correlation is given by:

$$(F * I)(x, y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} F(i, j).I(x+i, y+j)$$

We shall be using correlation to produce a blurred image from the original image with the help of a Gaussian kernel.

2.3 Padding:

Padding is simply a process of increasing the dimension of the image array by adding layers of zeros to it. The process of correlation reduces the dimension of the image array it is working on. Hence, in order to retain the size of the original image, it becomes necessary to pad it with outer layers of zeros.

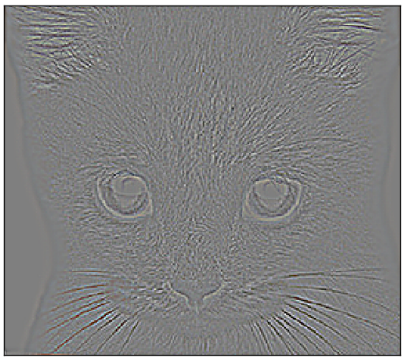
2.4 Creating & testing the filtering function:

To apply a filter on an image we have manually created a filter namely *my_imfilter* which imitates the functionality of the *cv2.filter2D* function in OpenCV. It takes an image and the filter as its inputs and returns the derived image after applying the filter. We have used this function in our *create_hybrid_image* function to create the low frequency and high frequency images and consequently the hybrid image by adding them. To test our function we have taken several filters (**identity filter, box filter, Sobel filter, Laplacian filter** etc.)

(a) Identity filter

(b) Box filter

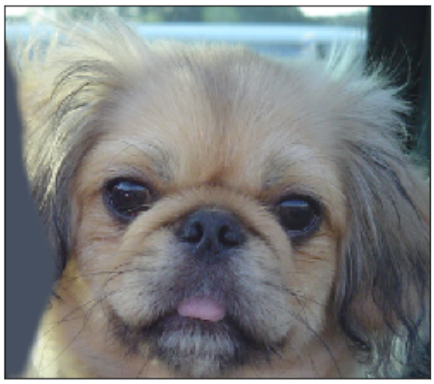
(c) Sobel filter



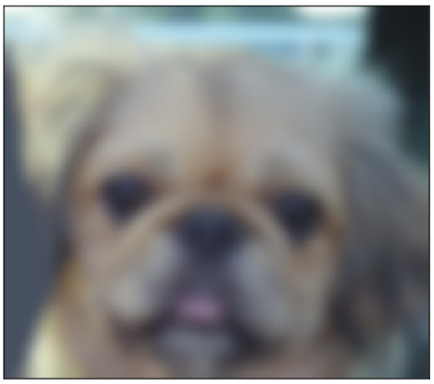
(d) Discrete Laplacian filter

2.5 Blurring the image:

We apply the Gaussian filter, or a low-pass filter over the padded image with the help of our $my\_imfilter()$ function which averages out the rapid changes in intensity i.e. it returns the blurred version of the original image.



(a) Original picture of dog



(b) Blurred picture of dog

2.6 Sharpening the image:

We get the sharpened image by subtracting the blurred image from the original one. This is equivalent to applying a high-pass filter over the original image.

(a) Original picture of cat

(b) Sharpened picture of cat

2.7 Adding two images:

We add the blurred version of one image and sharpened version of the other one to get our final hybrid image. Since we are not using OpenCV, so while adding the low and high frequency images it is possible that some values of the resultant matrix may go out of the range of the highest or the lowest pixel intensity values. To avoid such overflow we have to normalize the pixel values such that they remain in the range [0,255]

Figure 2.5: Hybrid image of dog and cat ($\sigma = 7$)

3 A few more examples:

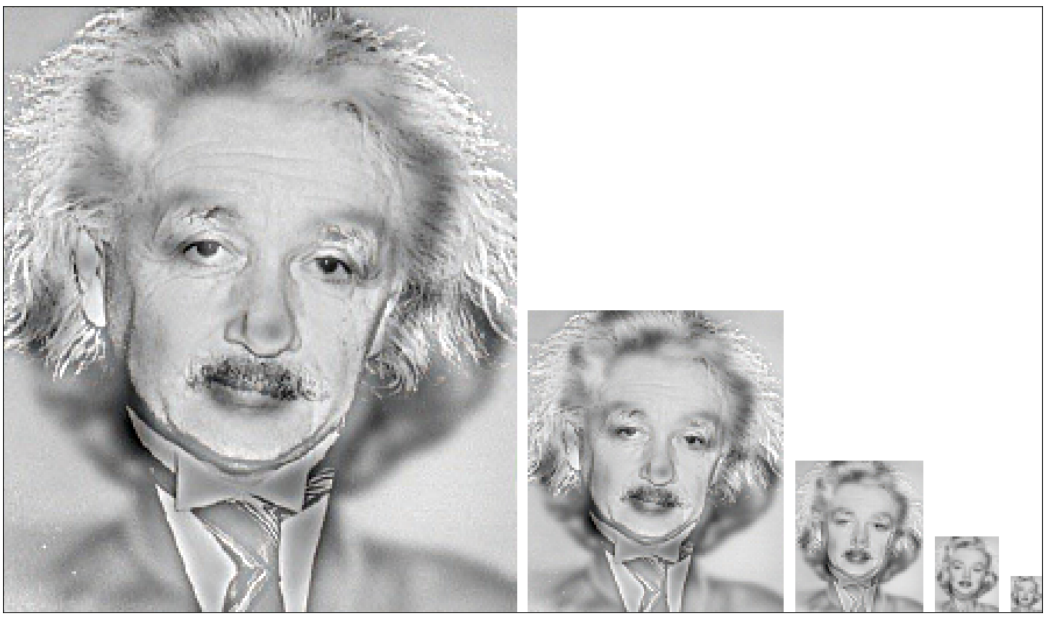


Figure 3.1: Hybrid image of Albert Einstein and Marilyn Monroe ($\sigma = 3$)

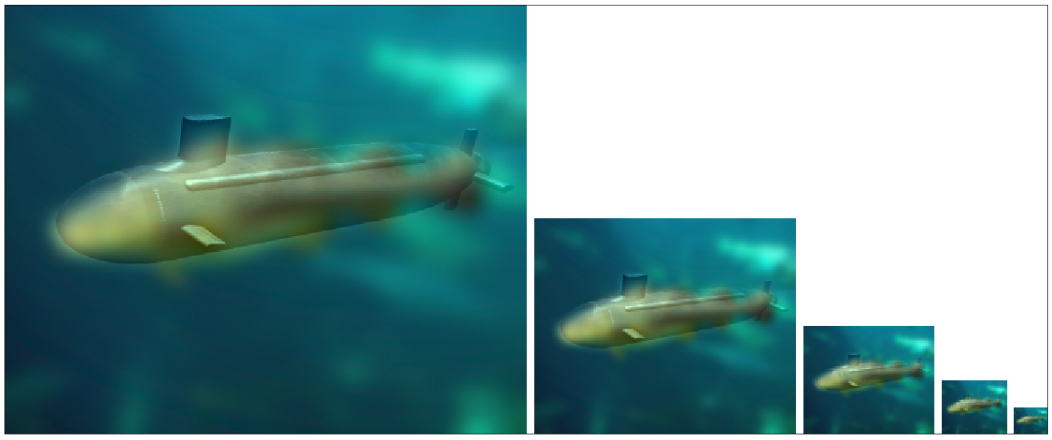


Figure 3.2: Hybrid image of a submarine and a fish ($\sigma = 7$)

Figure 3.3: Hybrid image of a bird and a plane ($\sigma = 7$)

Figure 3.4: Hybrid image of a bicycle and a motorcycle ($\sigma = 7$)

4 Conclusion:

In this project, we saw how our custom functions produced the same result as the OpenCV library. From the above hybrid image examples, we observed how different spatial frequencies have different interpretations according to viewing distance. We also saw that the hybrid image processing depends on the choice of the Gaussian filter. It was also surprising to see the limitations of human vision due to which we see the same image in two ways.

5 Acknowledgement:

We are highly grateful to Tamal Maharaj, Dept. of Computer Science, RK-MVERI for giving us the opportunity to work on this project. His guidance, supervision and support, throughout the project, has helped us immensely to complete it on time.