CS 815: Computer Vision Assignment-1

Creating hybrid images using Gaussian filter

Overview: The aim of this assignment is to make a program for creating hybrid images using Gaussian box filter. The primary reasoning for how we visualize a hybrid image is that the low-pass or low-frequency image is visible from a distance, since all its sharp features get blurred out. While all the sharp features are seen from nearby.

Hence for this particular part, we make a low frequency image, and a high frequency image by subtracting the low frequencies from the original image.

Special advice was given to not use any inbuilt function that skips steps to learn. Hence I've used a Gaussian box function derived from MATLAB.

MATLAB based Gaussian function code snippet

```
case 'gaussian' % Gaussian filter

     siz   = (p2-1)/2;
     std   = p3;

     [x,y] = meshgrid(-siz(2):siz(2),-siz(1):siz(1));
     arg   = -(x.*x + y.*y)/(2*std*std);

     h     = exp(arg);
     h(h<eps*max(h(:))) = 0;

     sumh = sum(h(:));
     if sumh ~= 0,
       h  = h/sumh;
     end;
```

Importing this to python:

```
import numpy as np

def matlab_style_gauss2D(shape=(3,3),sigma=0.5):
    """
    2D gaussian mask - should give the same result as MATLAB's
    fspecial('gaussian',[shape],[sigma])
    """
    m,n = [(ss-1.)/2. for ss in shape]
    y,x = np.ogrid[-m:m+1,-n:n+1]
    h = np.exp( -(x*x + y*y) / (2.*sigma*sigma) )
    h[ h < np.finfo(h.dtype).eps*h.max() ] = 0
    sumh = h.sum()
    if sumh != 0:
        h /= sumh
    return h
```

Setting Up:

Python based interpreter is required. Although it is suggested to use a py notebook, due to my unfamiliarity with python programming language, I used IntelliJ by Jet Brains IDE. So all the code was executed on my local machine. Also, it was easier, since the runtime was not too much. All the environment was already set up and ready to use.

Flow of program:

-Read the images, convert to matrix

-Calculate a Gaussian kernel matrix

-Multiply the Gaussian kernel to different parts of the image matrix, get summed values

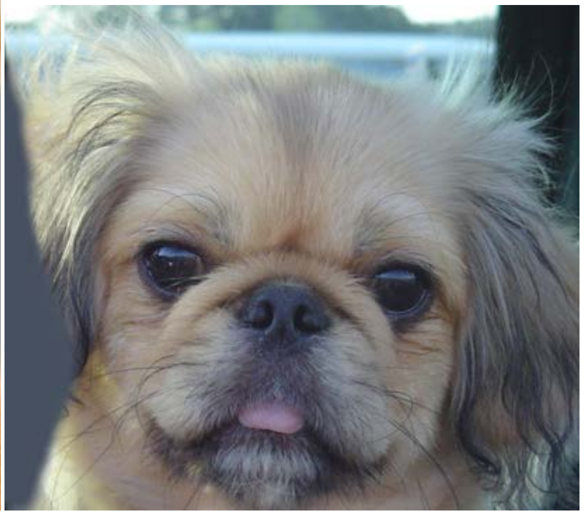
-Get a low frequency image, and get a high frequency image by subtracting lower frequency from the image.

-Combine to form a hybrid image

-Concatenate all images, while scaling down sizes for better observation

Data:

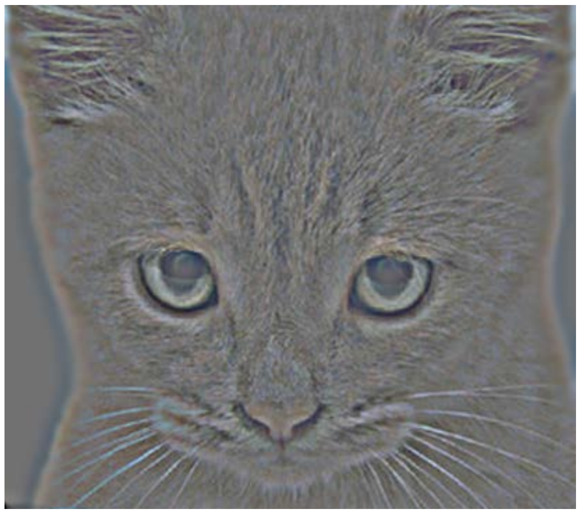
We were provided 10 images to work with. There was no specification on what image to use for low-pass and what to high-pass. Hence, I've selected by my own consideration.

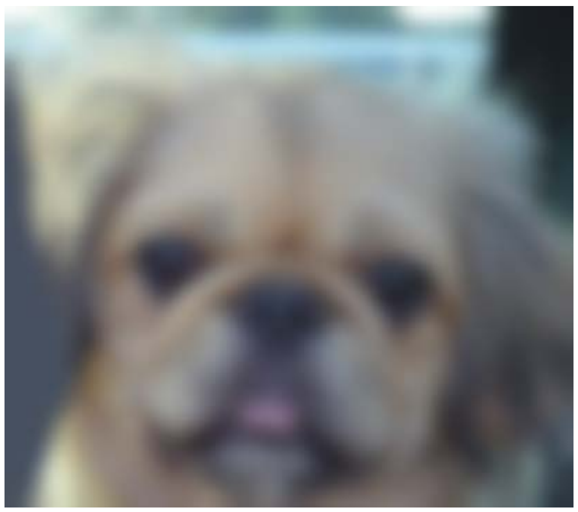


For example, the images of a cat and dog were taken for the 1st case.

For this case, I chose the image of dog, and converted it to a low frequency image by the Gaussian bix filter. Then the value of low frequency image was calculated for the cat image, and the values were deducted from the original image.

This left a low-pass image of the dog, and a high-pass image of the cat.





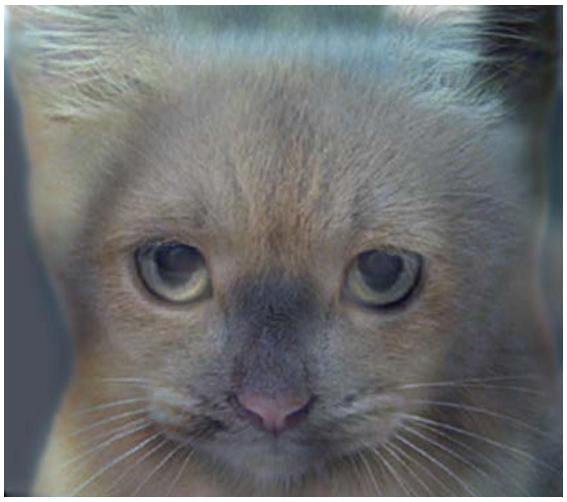
In the assignment, it was told to take specific value for sigma as sigma = 1.

I had adopted the code from a previously executed program, and hence I chose to take some different values for better results. The value of sigma was kept same as the cut off frequency (significantly higher). This allowed me to get a better blur effect.

However, the high-pass image I obtained was always different from the ones I saw online in the other available codes. Several different values were tried, but similar result was obtained each time.

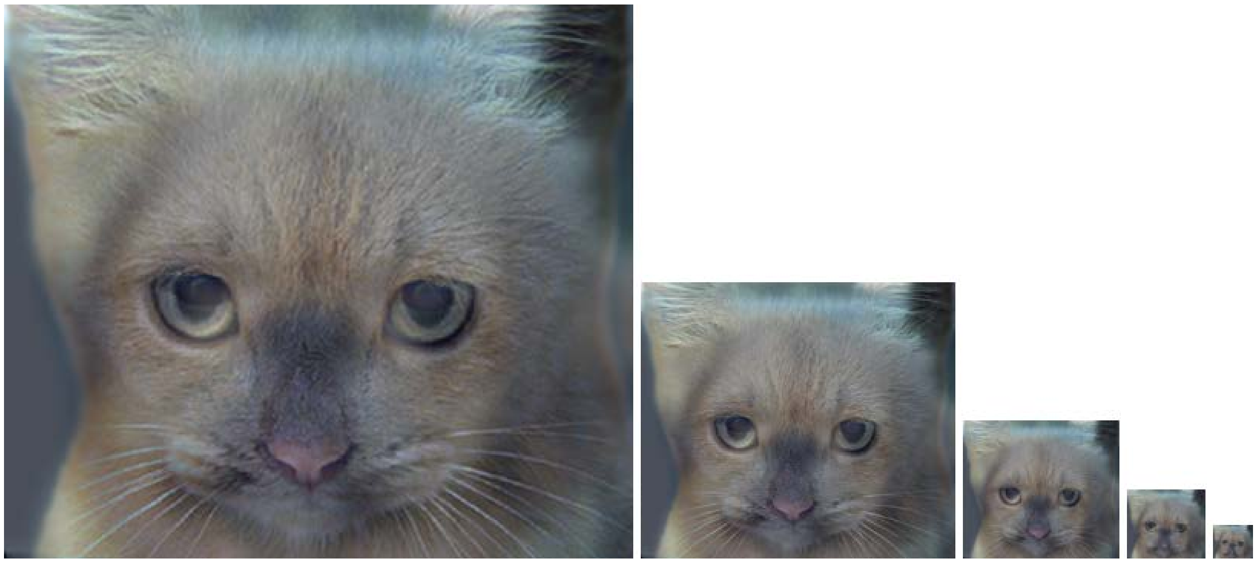
In the next step, the values of high-pass and low-pass images were added to create a superimposing sort of image which is also known as a hybrid image. In the image, the low pass image's features are blurred, hence it is easy to notice it from a distance. While the high-pass image can be noticed from a shorter distance easily.

Another way to check this is by holding a finger in front of the eye, focusing on it, and seeing the image with the peripheral vision. This adjusts the focus and the high frequency image is observed.



The results for both varied a lot with each set of images. Hence different values of cut off frequency were used for each image pair. For example, cut off frequency was taken as 7, 7 for the cat-dog pair, 5, 5 for the fish-submarine pair.

For each pair of image, it was told to make an image where the image would get downscaled. This would let the observer see the low-pass image from a closer distance, since all the sharp features of the high frequency image would get minimized due to the smaller size. And the blur features would appear.



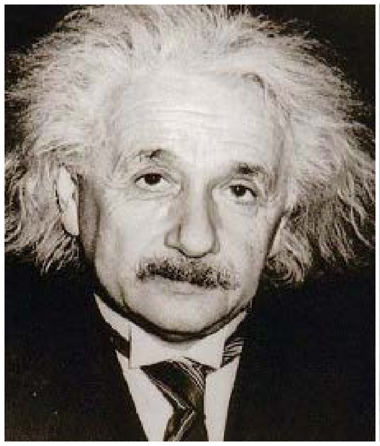
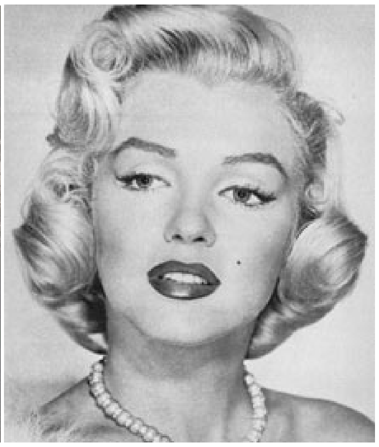
All the images were to be saved in the local directory.

Also, it was chosen to view all the images using a MATLAB plotter library that was imported in the python file.

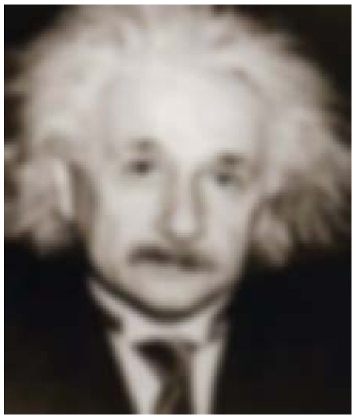
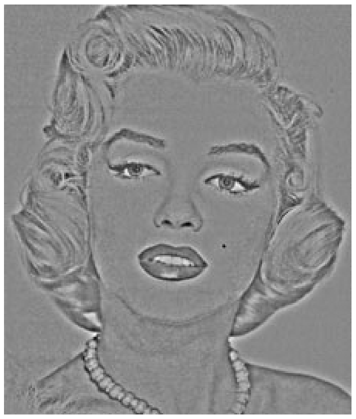
Other image pair results:
Einstein-Marilyn

Cut off frequency value- 3, 3

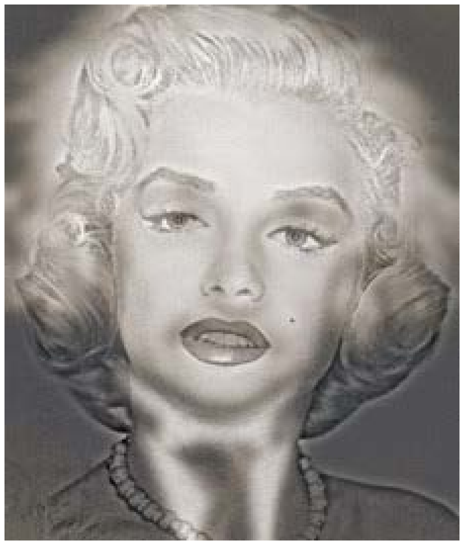
Images:

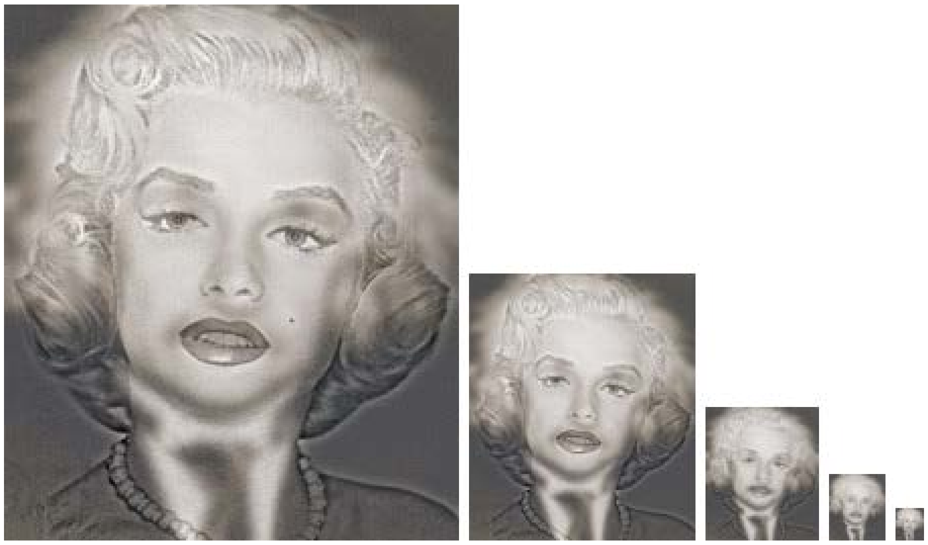
Low-pass, high-pass:

Hybrid Image:



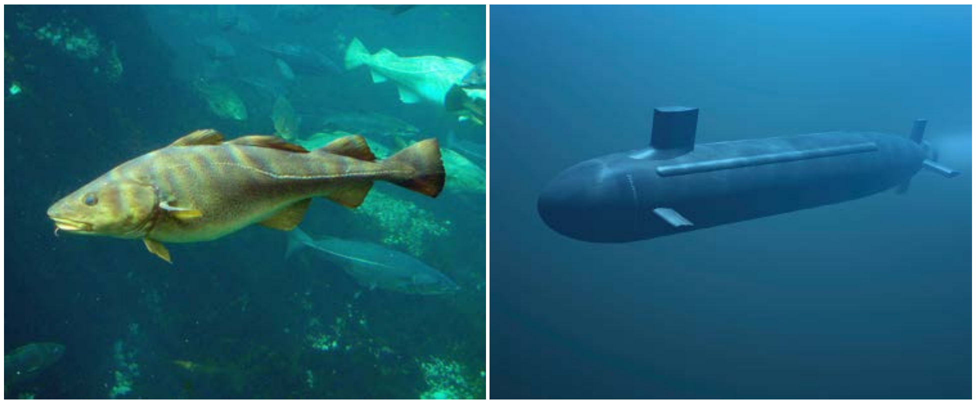
Scaled Hybrid Images:



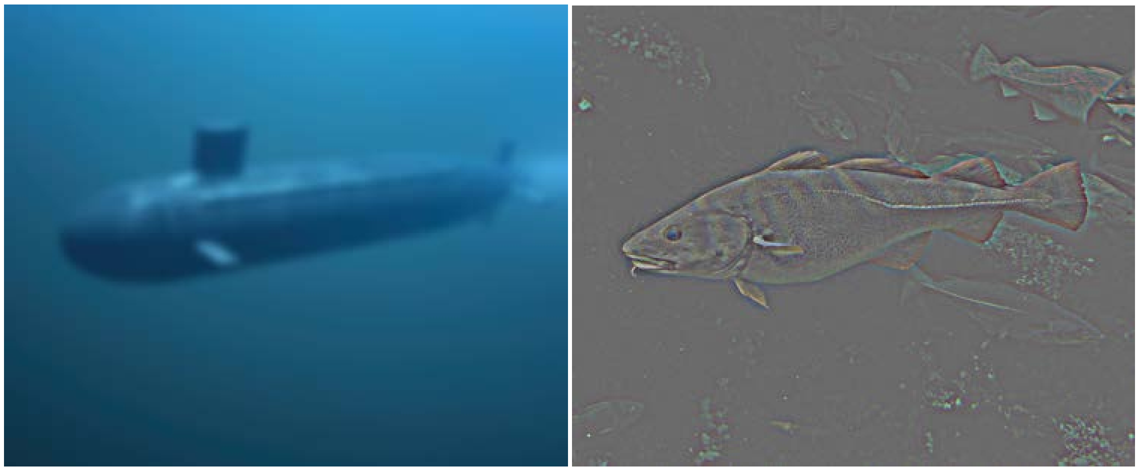
Fish-Submarine pair

Cut off frequency value- 4, 4

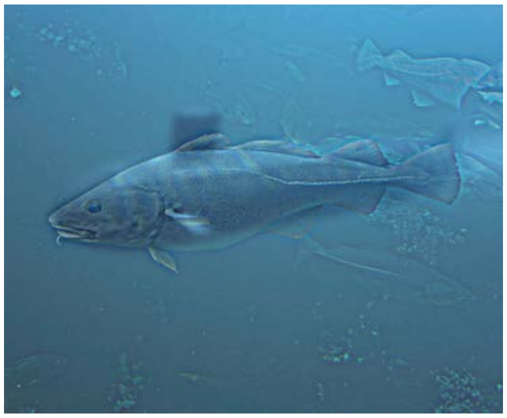
Images:



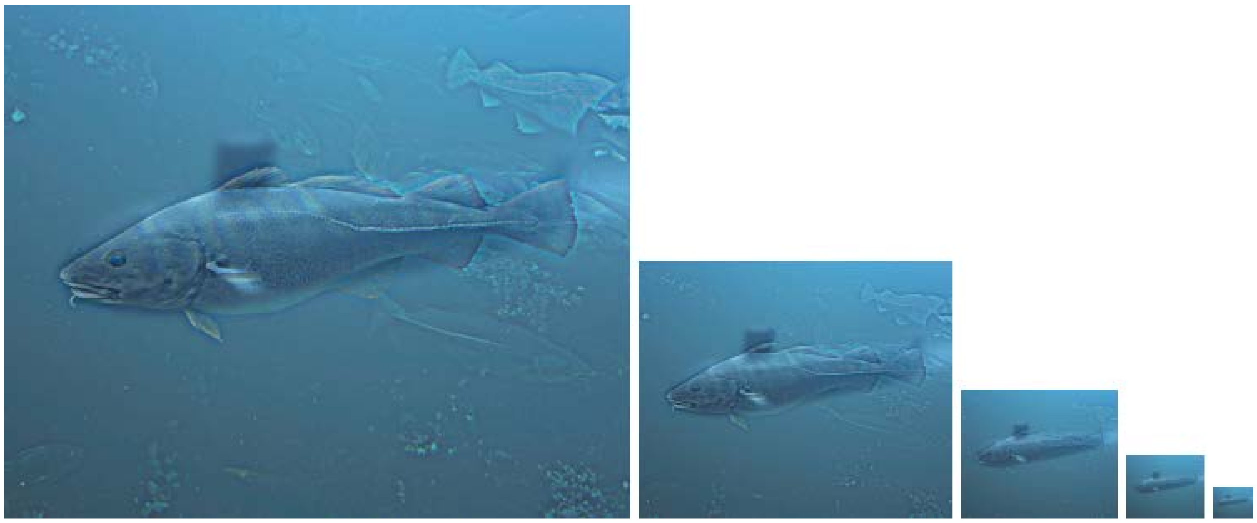
Low-pass, high-pass:



Hybrid Image:



Scaled Hybrid Images:

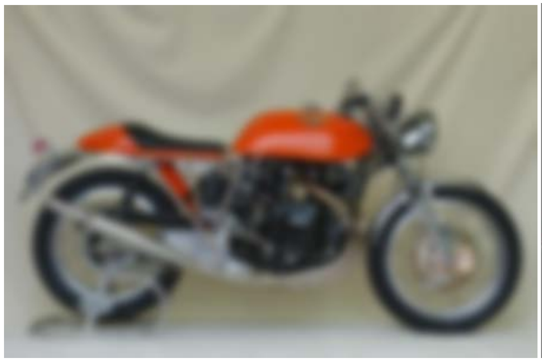
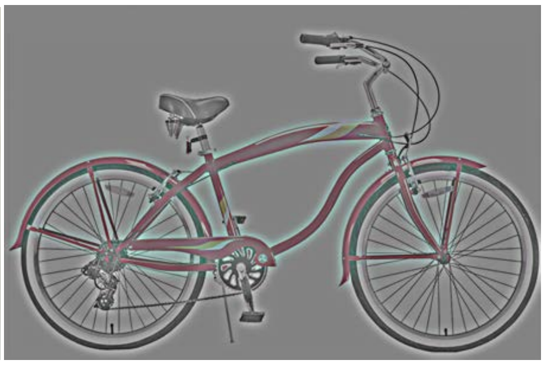


Bicycle-Motorcycle pair

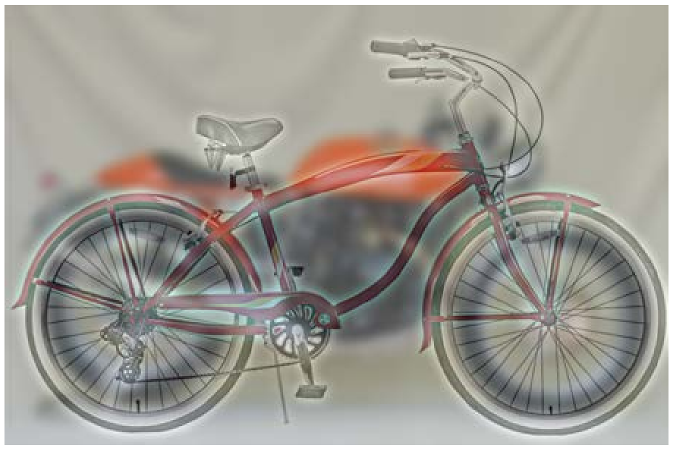
Cut off frequency value- 4, 4

Images:

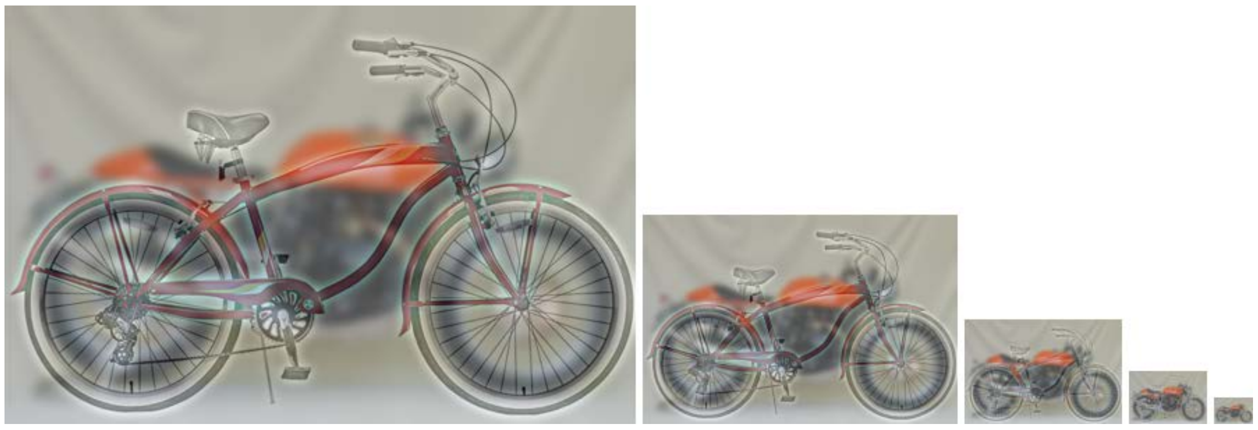
Low-pass, high-pass:

Hybrid Image:



Scaled Hybrid Images:

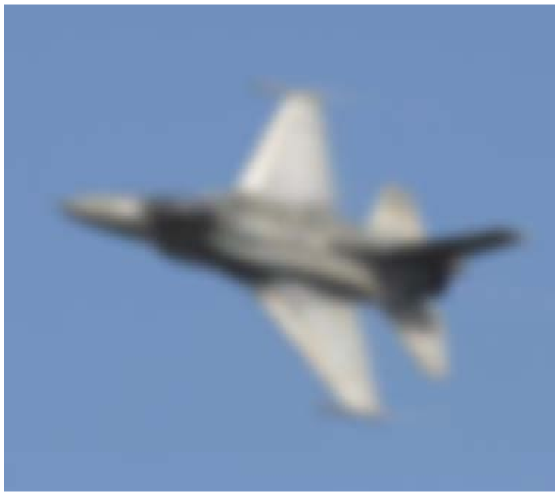
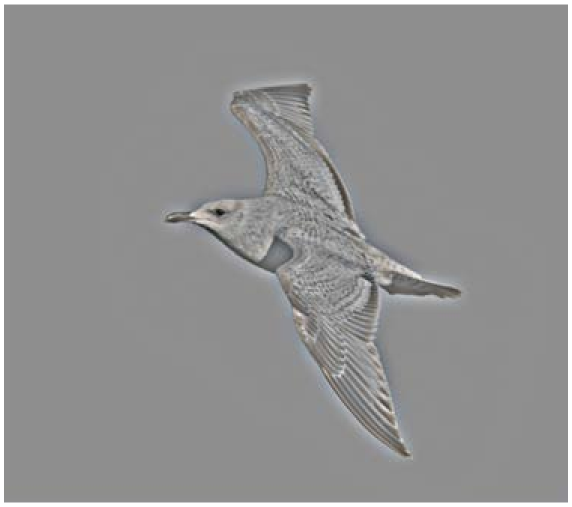


Bird-Plane pair:

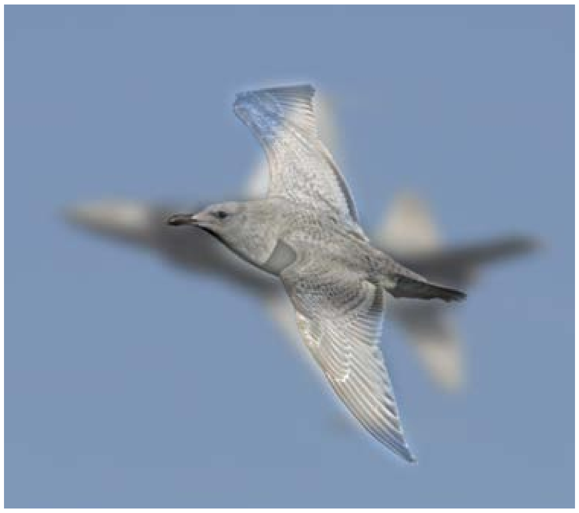
Cut off frequency value- 5, 5

Images:

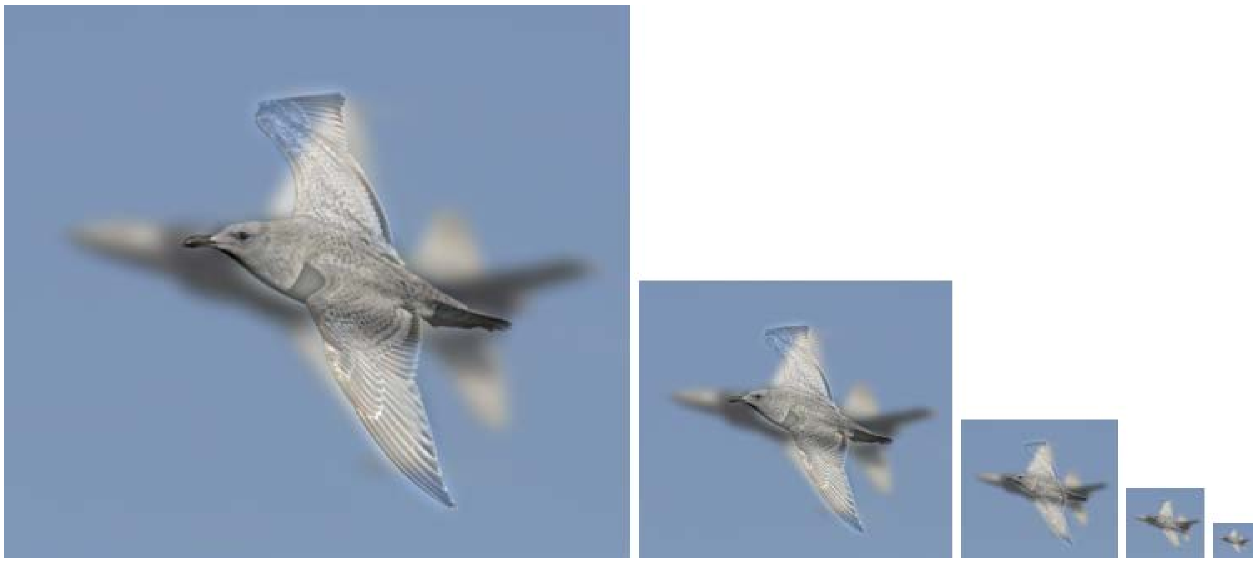
Low-pass, high-pass:



Hybrid Image:



Scaled Hybrid Images:



The results obtained for all were satisfying, but the bicycle-motorcycle pair didn't go well together. While for the bird-plane pair, the size of the airplane made the hybrid image appear a bit weird. The sharp features of the bird do not cover the entire portion of the plane, hence the perceivable discrepancy.

Other filters:

It wasn't specifically mentioned as to what filters to use, and hence a few filters were used for this assignment.

For the application, the image of cat was used.

Identity filter:

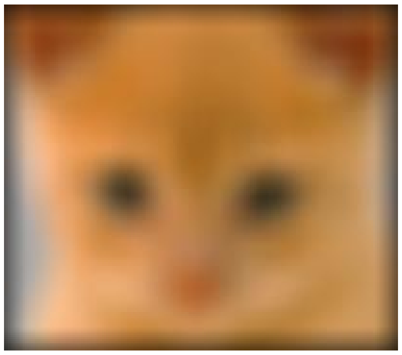
The box filter used is an identity matrix, hence there will be no effect on the image.

Small blur filter:

Blurs the image but to a smaller extent

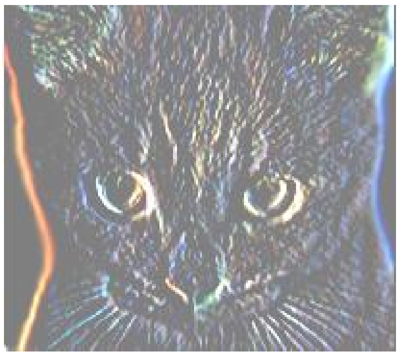
Large Blur filter:

Blurs to a larger extent. OpenCV Gaussian function was called since this was for demonstration.



Sobel Edge Filter:

This is a sort of horizontal filter, hence it removes all the horizontal edges in the image



Credits:

MATLAB function:

Link to the function.