ADVANCED SIGNAL PROCESSING - ASSIGNMENT 2

KOMMINENI ADITYA - EE17B047

I FILTERING :

1. Each of the filtering operations have been implemented in the .ipynb file and the corresponding plot for each of the filtered signal have been plotted.

2. For each of the operation below is the result on whether they are linear and space invariant or not.

   a. Yes the operation is linear space invariant.

$$y_k = x_{k+1} - x_k$$

if input is $x' = ax$

$$y'_k = x'_{k+1} - x'_k = a(x_{k+1} - x_k) = a y_k$$

$$y^1_k = x^1_{k+1} - x^1_k$$

$$y^2_k = x^2_{k+1} - x^2_k$$

$$x' = x^1 + x^2$$

$$y'_k = x^1_{k+1} + x^2_{k+1} - (x^1_k + x^2_k)$$

$$= (x^1_{k+1} - x^1_k) + (x^2_{k+1} - x^2_k)$$

$$= y^1_k + y^2_k$$

$\therefore$ linear

let x_{k-n} be the input

$$y'_k = x_{k-n+1} - x_{k-n}$$

$$y_{k-n} = x_{k-n+1} - x_{k-n}$$

$$y'_k = y_{k-n}$$

$\therefore$ space invariant

b. No, the operation is not linear space invariant

$$y_k = x_k - \bar{x} \; ; \; \bar{x} = \frac{1}{L+1}\sum_{0}^{L} x_i$$

let input be $x'_k = a x_k$

$$y'_k = a x_k - \bar{x} \neq a y_k$$

$\therefore$ not linear

c. No, the operation is not linear space invariant

$$y_k = \text{median}(\{x_l : l \in [k-2, k+2]\})$$

let $x'_k = x^1_k + x^2_k$

$$y'_k = \text{median}(\{x'_k : l \in \{k-2, k+2\})$$

$$= \text{median}(\{x^1_k + x^2_k : l \in \{k-2, k+2\})$$

$$\neq \text{median}(\{x^1_k\}) + \text{median}(\{x^2_k\})$$

$\therefore$ not linear

d. Yes, the operation is linear space invariant

$$y_k = x_{k+0.5} - x_{k-0.5}$$

$$= \frac{x_{k+1} - x_{k-1}}{2}$$

let $x'_k = x^1_k + x^2_k$

$$y'_k = \frac{x'_{k+1} - x'_{k-1}}{2} = \frac{x^1_{k+1} - x^1_{k-1} + x^2_{k+1} - x^2_{k-1}}{2}$$

$$= y^1_k + y^2_k$$

set $x'_k = a x_k$

$$y'_k = x'_{k+1} - x'_{k-1} = a(x_{k+1} - x_{k-1})$$

$$= a y_k$$

set $x'_k = x_{k-n}$

$$y'_k = x'_{k+1} - x'_{k-1} = x'_{k-n+1} + x'_{k-n-1}$$

$$= y_{k-n}$$

$\therefore$ linear & space invariant

e. No, the operation is not linear space invariant

$$y_k = |x_{k+0.5} - x_{k-0.5}|$$

$$= \frac{|x_{k+1} - x_{k-1}|}{2}$$

let $x'_k = x^1_k + x^2_k$ be the input

$$y'_k = \frac{|x'_{k+1} - x'_{k-1}|}{2} = \frac{|x^1_{k+1} - x^1_{k-1} + x^2_{k+1} - x^2_{k-1}|}{2}$$

$$\neq \frac{|x^1_{k+1} - x^1_{k-1}|}{2} + \frac{|x^2_{k+1} - x^2_{k-1}|}{2}$$

∴ not linear

f. Yes, the operation is linear space invariant

$$y_k = \frac{1}{5} \sum_{i=k-2}^{i=k+2} x_i$$

let $x'_k = a x^1_k + b x^2_k$

$$y'_k = \frac{1}{5} \sum_{i=k-2}^{i=k+2} x'_i = \frac{1}{5} \sum_{i=k-2}^{i=k+2} (a x^1_i + b x^2_i)$$

$$= \frac{a}{5} \sum_{i=k-2}^{i=k+2} x^1_i + \frac{b}{5} \sum_{i=k-2}^{i=k+2} x^2_i$$

$$= a y^1_k + b y^2_k$$

∴ linear

let $x'_k = x_{k-n}$

$$y'_k = \frac{1}{5} \sum_{i=k-2}^{i=k+2} x_i = \frac{1}{5} \sum_{i=k-2}^{k=k+2} x_{i-n}$$

$$= \frac{1}{5} \sum_{j-n=k-2}^{j-n=k-2} x_j \quad \{j = i-n\}$$

$$= \frac{1}{5} \sum_{j=k-n-2}^{j=k-n+2} x_j = y_{k-n}$$

∴ space invariant

3. From the above, we can see that a,d,f are linear space invariant. In order to obtain the final result of the convolution, we apply the filters below by inverting them and sliding them along the signal. In order to facilitate the process, we need to zero pad the signal accordingly. The convolution filters corresponding to all the three are as follows :

a. Filter = [1, -1 , 0]

d. Filter = [½ , 0 , -½]

f. Filter = [⅕ , ⅕ , ⅕ , ⅕ , ⅕]

4. The results of the final signal y_k obtained by convolution and that obtained by direct implementation coincide with each other as shown in the graphs in ipynb file.

II FILTERING IN FOURIER SPACE :

1. The filters have been implemented in the fourier domain employing the np.fft.fft() function which converts the given signal into its FFT and then we can convert the frequency domain to space domain using the np.fft.ifft() function.

We know that in spacial domain if we are convolving two signals then in the frequency domain, we must multiply the two signals.

2. If we take the FFT of the 16 point signal and the corresponding filters, then we will not get the exactly similar output signal as obtained while convolving. The reason is because if we were to do circular convolutions in order to find the DFT, then we would be needed to pad the signals with zeros in order to obtain the proper DFT function. Hence, if we want the convolution value to match, then we would have to pad the spacial domain signal and the perform the FFT and multiply with the convolution filter FFT. Therefore, in order to match, we have zero padded the signal i.e. zero padded the signal with the same number as the size of the filter - 1.

3. Once we do the above, we can see that the results match exactly as given by the plots in the ipynb file.

III HYBRID IMAGES :

IMPLEMENTATION DETAILS :

In order to deal with the images wherein there is discrepancy of size, we resize both the images to mean of the dimensions. I choose this over resizing them to either one of the dimensions because resizing them to either one of the images original dimensions might result in complete distortion of one image which may be undesirable.

We apply the gaussian filtering over all dimensions of the image i.e. if the image is RGB then we apply the gaussian individually for R, G and B.

We experiment with various values of sigma before sticking to a given value which would provide the best results.

In order to obtain th results of the pyramidal images, please run the helpers.py with the appropriate file names.

Below are the details obtained for all the 7 sets of images given in the folder.

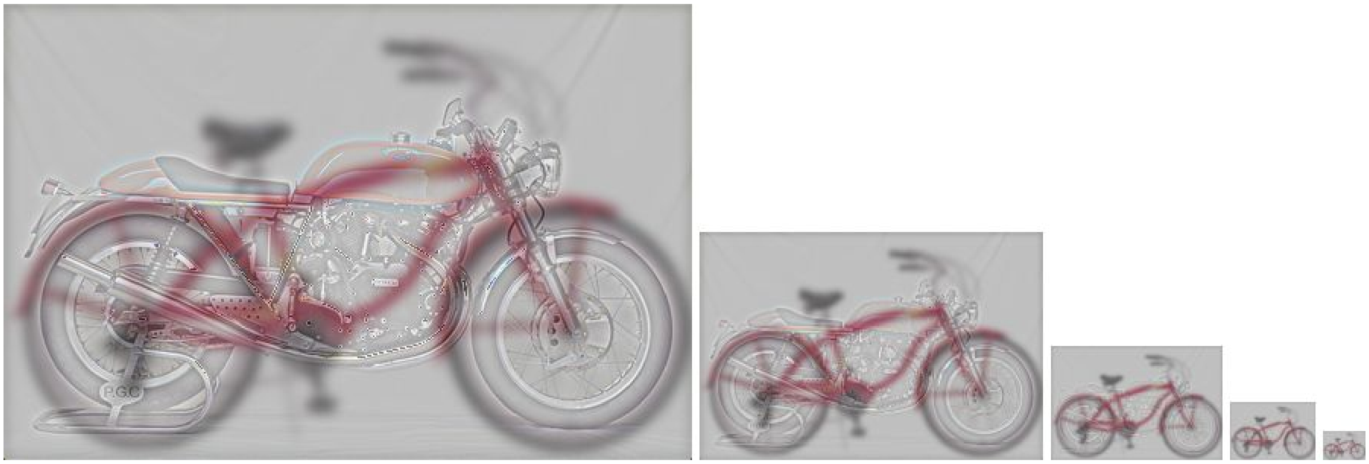
<u>ex01:</u>

Below are the two images of pair :

The bicycle on the left and the motorcycle on the right. In order to generate the hybrid image, we consider the bicycle to be the image on which we implement the low pass filtering and the motorcycle to be the image on which we implement the highpass filtering.

The sigma for the motorcycle is 10 and that for bicycle is 15.

The hybrid image for different scales generated using the helper.py function is as given below :



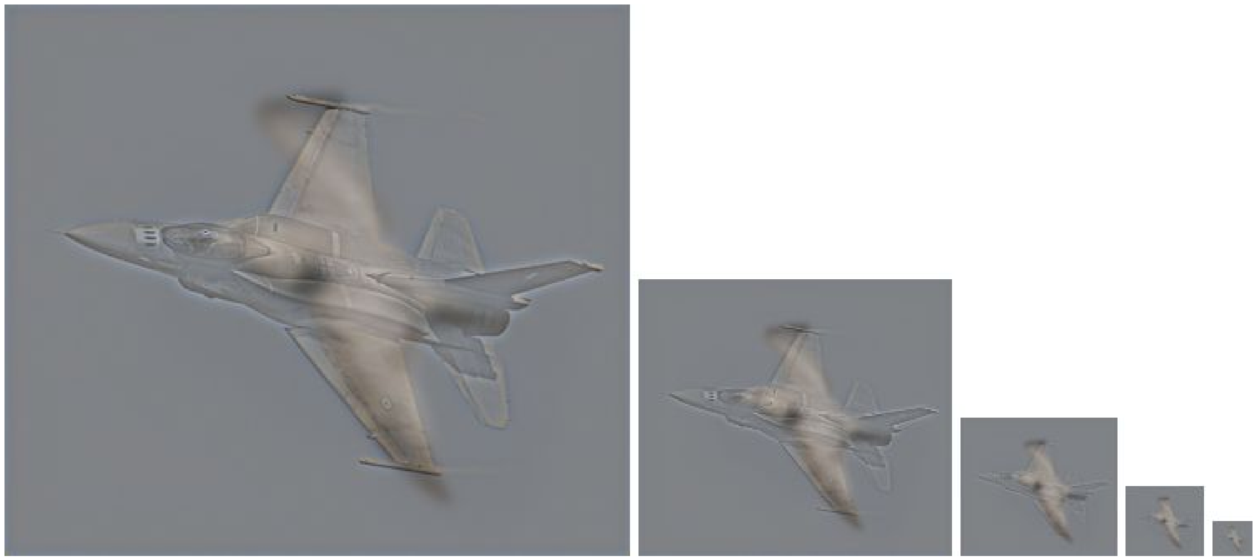
ex02:

Below are the two images of pair :

The plane on the left and the bird on the right. In order to generate the hybrid image, we consider the bird to be the image on which we implement the low pass filtering and the plane to be the image on which we implement the highpass filtering.

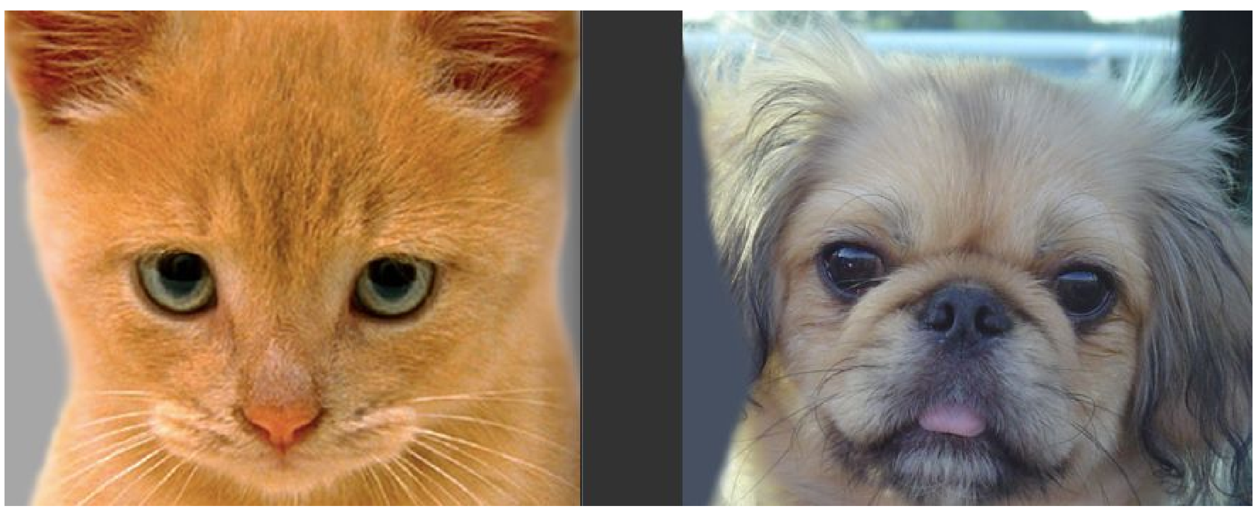
The sigma for the plane is 10 and that for bird is 15.

The hybrid image for different scales generated using the helper.py function is as given below :



ex03:

Below are the two images of pair :



The cat on the left and the dog on the right. In order to generate the hybrid image, we consider the dog to be the image on which we implement the low pass filtering and the cat to be the image on which we implement the highpass filtering.

The sigma for the cat is 30 and that for dog is 20.

The hybrid image for different scales generated using the helper.py function is as given below :

ex04:

Below are the two images of pair :

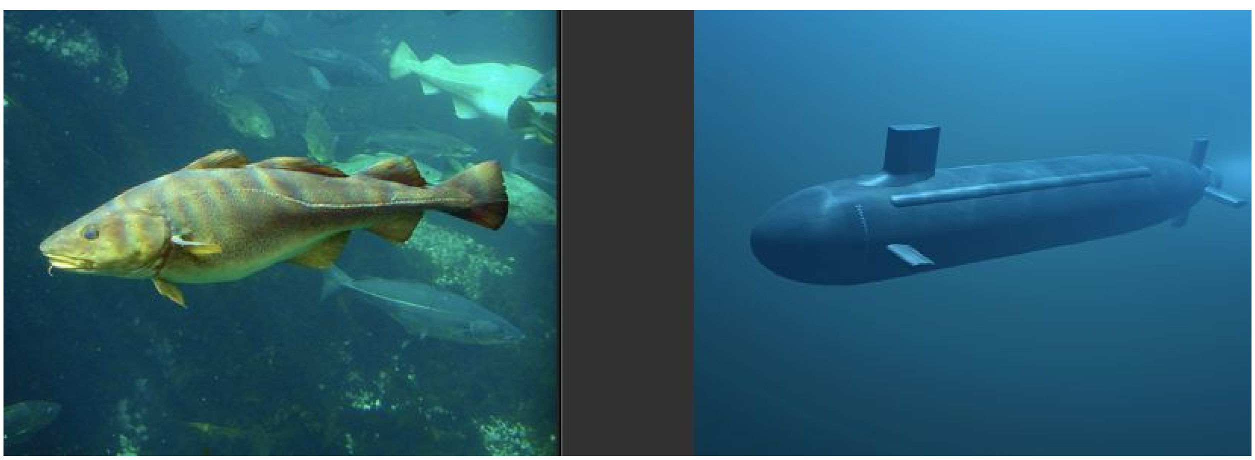
Einstein bicycle on left and Marilyn on right. In order to generate the hybrid image, we consider Einstien to be the image on which we implement the low pass filtering and Marilyn to be the image on which we implement the highpass filtering.

The sigma for Einstein is 10 and that for Marilyn is 10.

The hybrid image for different scales generated using the helper.py function is as given below :

ex05:

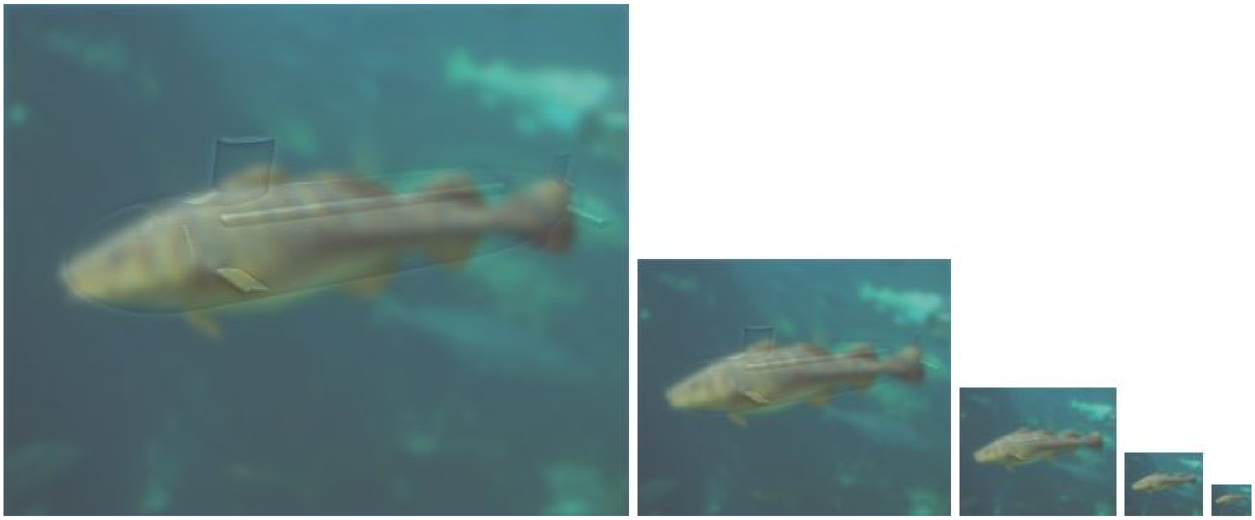
Below are the two images of pair :



Fish on left and submarine on right. In order to generate the hybrid image, we consider fish to be the image on which we implement the low pass filtering and submarine to be the image on which we implement the highpass filtering.

The sigma for fish is 10 and that for submarine is 10.

The hybrid image for different scales generated using the helper.py function is as given below :



The reason we don't see the submarine in this case and only can see the outline could be owing to the fact that the colour of submarine and the background are almost identical to each other owing to which only the edges in the submarine could have been left behind in the high pass filtering.

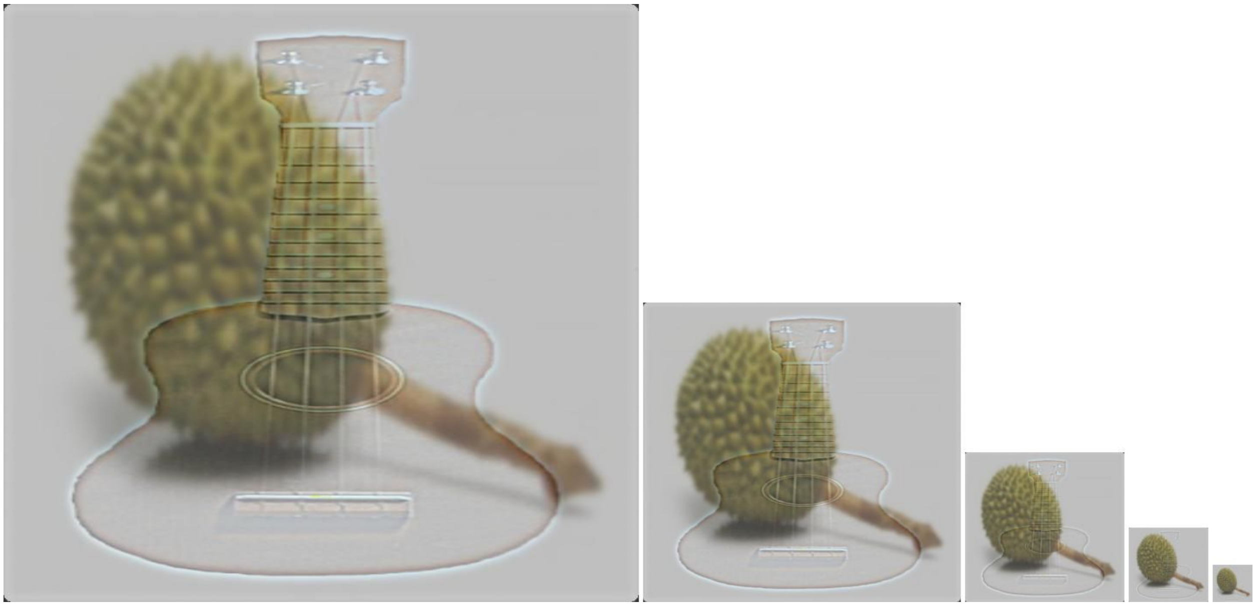
ex06:

Below are the two images of pair :

Durian on left and Ukulele on right. In order to generate the hybrid image, we consider durian to be the image on which we implement the low pass filtering and the ukulele to be the image on which we implement the highpass filtering.

The sigma for durian is 18 and that for ukulele is 27.

The hybrid image for different scales generated using the helper.py function is as given below :



In the above picture, the original size of both the images is non uniform owing to which there is slight distortion in both the images in the final recreated image.

ex07:

In this example, both the images do not have the same size and the representations below arent of the same size. They have been resized to show the images. Below are the two images of pair :

Einstein on left and Marilyn on right. In order to generate the hybrid image, we consider Einstein to be the image on which we implement the low pass filtering and Marilyn to be the image on which we implement the highpass filtering.

The sigma for the Einstein is 19 and that for Marilyn is 28.

The hybrid image for different scales generated using the helper.py function is as given below :

OBSERVATIONS:

From the above images, we can see that it is possible to perceive the same image as two different worldly objects at different scales.

The role of sigma is extremely important when deciding whether the hybrid image is generated as intended or not. Wrong values of the sigmas could result in no effective separation between two distinct images being seen at different scales.

If the images are of different scale, then resizing them and calculating the hybrid images do not seem to severely disturb the hybrid image unless the image is extremely distorted such that much of the information in the image is lost.