**CSci 5561 – Homework 2**

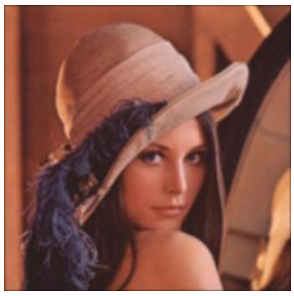
**Rohith Nallamaddi**

1. **Description:** Gaussian pyramid and Laplacian pyramids of the input images are obtained from getPyr.m
   **Implementation steps:** To get Gaussian pyramid we smoothen and down sample the image by a factor of 2 at each level of the pyramid. We down sample the image because, as we are scaling down the image in spatial domain, the frequency spectrum gets stretched in the frequency domain which may lead to overlap of adjacent spectrums and lead to aliasing. Therefore, to prevent aliasing effect we smoothen the image before down sampling.
   Similarly, for Laplacian pyramid construction up sampling is involved which will compress the spectrum in frequency domain, smoothing after up sampling will select the relevant part of the spectrum. The Laplacian pyramid essentially stores the high frequency components at different scales as it is the difference between two Gaussians.
   The filter kernel H is multiplied by a factor of 4 while smoothing after up sampling because the range of pixels it covers is same but the alternate pixels are zero. To embed more neighborhood information at the current pixel by increasing their weights, we multiply the filter by a factor of four.

   **Test Results:**

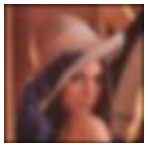
   **Lena - Gaussian Pyramid:**
   Level 1:

   Level 2:



Level 3:

Level 4:

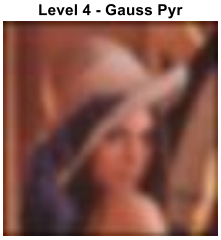


Level 1 - Gauss Pyr

Level 2 - Gauss Pyr

Level 3 - Gauss Pyr

Level 4 - Gauss Pyr



**Lena – Laplace Pyramid:**

Level 1:

Level 2:

Level 3:

Level 4:


Level 1 - Laplace Pyr
Level 2 - Laplace Pyr
Level 3 - Laplace Pyr
Level 4 - Laplace Pyr

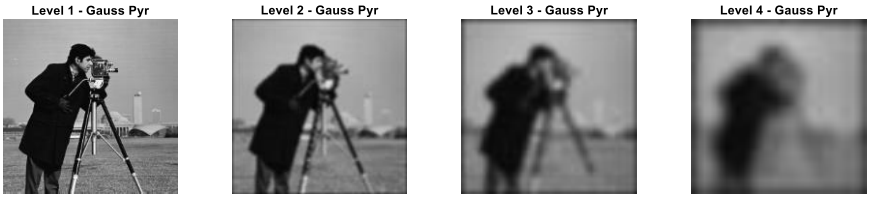
**Cameraman – Gaussian Pyramid:**

Level 1:

Level 2:

Level 3:

Level 4:


Level 1 - Gauss Pyr
Level 2 - Gauss Pyr
Level 3 - Gauss Pyr
Level 4 - Gauss Pyr

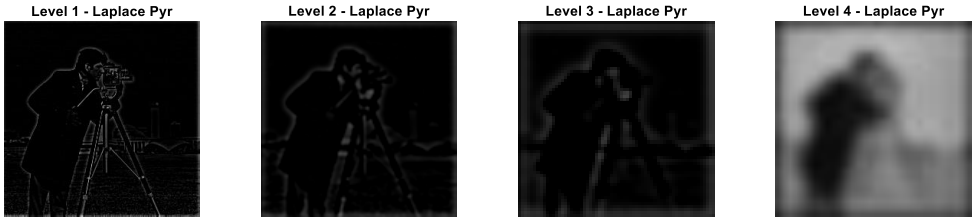
**Cameraman – Laplacian Pyramid**

Level 1:

Level 2:

Level 3:

Level 4:

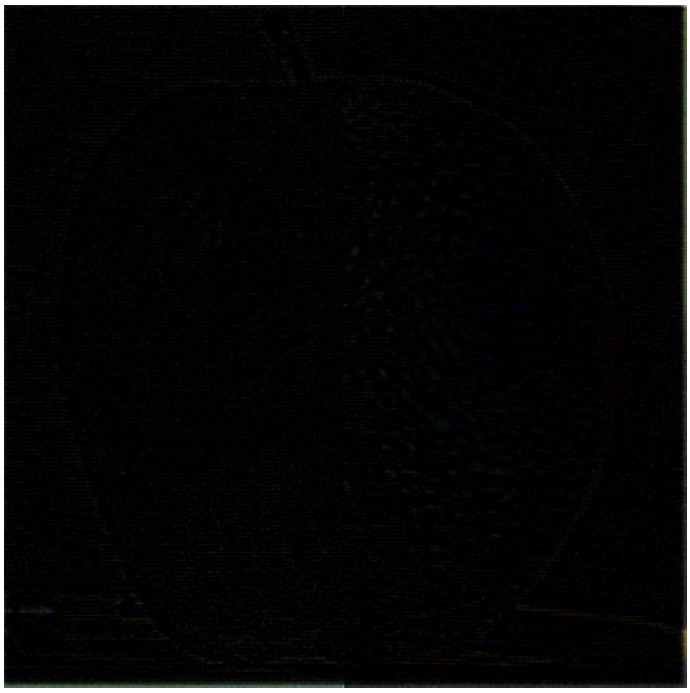

Level 1 - Laplace Pyr
Level 2 - Laplace Pyr
Level 3 - Laplace Pyr
Level 4 - Laplace Pyr

2. **Description:**

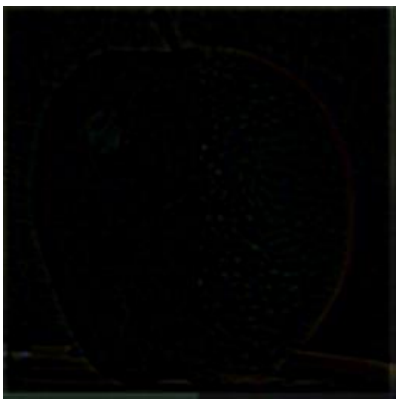
In this task, the concept of multi scale representation of images has been leveraged to blend two images. While splicing the apple and orange images together along the midline produces a noticeable cut, splining them together creates a beautiful illusion of a truly hybrid fruit. The key to this approach is that the low-frequency colour variations between the red apple and the orange are smoothly blended, while the higher-frequency textures on each fruit are blended more quickly to avoid "ghosting" effects when two textures are overlaid. That is, the low-level frequencies are blended first at higher levels of the pyramid and high frequency components are blended later. The Gaussian pyramid of the mask used plays a crucial role in blending two images smoothly by selecting relevant portion of the two images and mix at the edge.

**Results:**

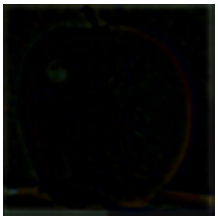
Laplace pyramid level 1:



Laplace pyramid level 2:



Laplace pyramid level 3:



Laplace pyramid level 4:

Laplace pyramid level 5:

Laplace pyramid level 6:

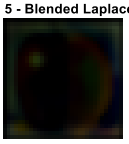
Level 1 - Blended Laplace Pyr

Level 2 - Blended Laplace Pyr

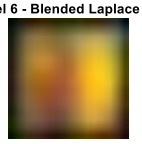
Level 3 - Blended Laplace Pyr

Level 4 - Blended Laplace Pyr

Level 5 - Blended Laplace Pyr



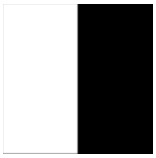
Level 6 - Blended Laplace Pyr



Gaussian Mask 1 – Gaussian Pyramid:

Level 1 - Gaussian Mask1 Pyr

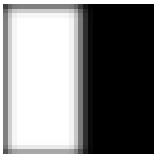
Level 2 - Gaussian Mask1 Pyr



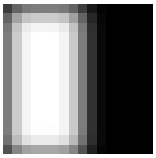
Level 3 - Gaussian Mask1 Pyr

Level 4 - Gaussian Mask1 Pyr

Level 5 - Gaussian Mask1 Pyr



Level 6 - Gaussian Mask1 Pyr



Gaussian Mask 2 – Gaussian Pyramid:

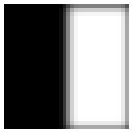
Level 1 - Gaussian Mask2 Pyr

Level 2 - Gaussian Mask2 Pyr

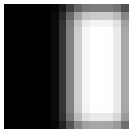
Level 3 - Gaussian Mask2 Pyr

Level 4 - Gaussian Mask2 Pyr

Level 5 - Gaussian Mask2 Pyr



Level 6 - Gaussian Mask2 Pyr



Bonus Question: To blend the hand and eye, a suitable mask has to be chosen around the eye which has 1’s only in the region of eye and zeros at the rest of the image. The inverse of this mask has to be applied on the hand. This way we can select the required features from two images. After modifying the mask for this specific application, same procedure as above can be applied to blend the images.

Result:

Final Blended image

Mask used on the eye image:

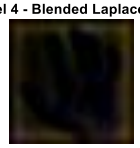
Laplace pyramid of blended image:

Level 1 - Blended Laplace Pyr

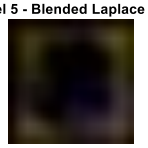
Level 2 - Blended Laplace Pyr

Level 3 - Blended Laplace Pyr

Level 4 - Blended Laplace Pyr



Level 5 - Blended Laplace Pyr



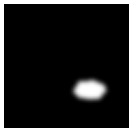
Level 6 - Blended Laplace Pyr

Gaussian Pyramid of Mask used on eye:

Level 1 - Gaussian Mask2 Pyr

Level 2 - Gaussian Mask2 Pyr

Level 3 - Gaussian Mask2 Pyr



Level 4 - Gaussian Mask2 Pyr

Level 5 - Gaussian Mask2 Pyr

Level 6 - Gaussian Mask2 Pyr

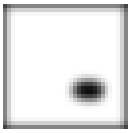
Gaussian Pyramid of Mask used on hand:

Level 1 - Gaussian Mask1 Pyr

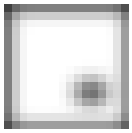
Level 2 - Gaussian Mask1 Pyr

Level 3 - Gaussian Mask1 Pyr

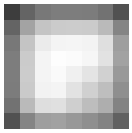
Level 4 - Gaussian Mask1 Pyr



Level 5 - Gaussian Mask1 Pyr



Level 6 - Gaussian Mask1 Pyr

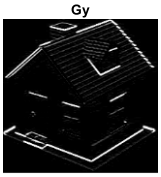


3. **Results**:
   **House**.jpg:

Original

Gx

Gy



Magnitude

Direction

**Cameraman.jpg:**


The normalization factor for the kernel is 1/6 which is the total number of non-zero values in the kernel.

The Prewitt operator can be extended to 5x5 kernels as well. The role of Prewitt kernel is to emphasize horizontal gradient by emphasizing vertical edges. The transpose of the kernel can be used to perform emphasize the vertical gradient. The 5x5 kernel can be written as below to compute X and Y gradients

| | | | | |
|---|---|---|---|---|
| 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 |
| -1 | -1 | -1 | -1 | -1 |
| -2 | -2 | -2 | -2 | -2 |

| | | | | |
|---|---|---|---|---|
| -2 | -1 | 0 | 1 | 2 |
| -2 | -1 | 0 | 1 | 2 |
| -2 | -1 | 0 | 1 | 2 |
| -2 | -1 | 0 | 1 | 2 |
| -2 | -1 | 0 | 1 | 2 |

The rationale behind choosing such kernel is that, it is based on the idea of contrast difference. Prewitt kernel emphasizes the weights of the neighborhood pixels as the distance from the centre increases. That is the weight is proportional to the distance from the centre.

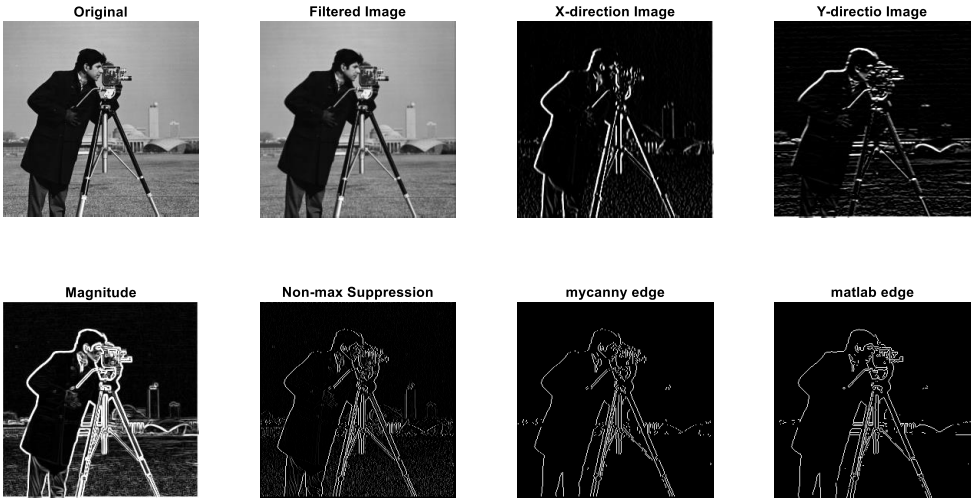
4. **Description:**

**Canny implementation:**

In the first step image is smoothened using Gaussian 3x3 filter to reduce the noise as noise can be wrongly interpreted as edge in the image. I chose a default standard deviation value of 0.5 for smoothening kernel as it moderately weighs the neighbourhood pixels. This value was found empirically by trial and error and seemed to give good results.

In the second step, Prewitt kernel is used to find the gradient magnitude and orientation.
Un-normalized version of the Prewitt kernel has been used as we can ignore normalization factor for edge detection. We can ignore normalization factor because it scales magnitude of all the edge pixels equally and we can choose a appropriate threshold value which is an hyper parameter set by us to tune the performance of the edge detector.
I considered 8 different directions of edges each with an angle of 45 degrees
(-22.5 to 22.5 -> horizontal right, 22.5 to 67.5 -> diagonal etc. each direction with an increment of 45 degrees)
I chose high and low resolution thresholds as a fraction of maximum value of my non-maxima suppressed image. I chose high threshold value to be around 0.25~0.3 of maxima of suppressed image and maintained constraint that ration of high to low threshold to be in the range of 2~3.
The role of high threshold is to detect strong edge and start the edge point to be built upon. The role of low threshold is to detect weak edges which are connected to the strong edges.

**Results:**
**Cameraman:**

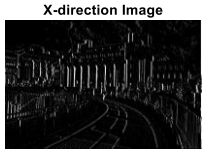



**Coffman**

Original

Filtered Image

X-direction Image

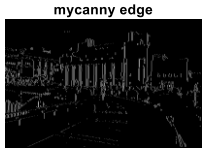


Y-directio Image

Magnitude

Non-max Suppression

mycanny edge



matlab edge