Problem Details: You will practice the programming of Gaussian pyramid and Laplacian pyramid. Both pyramids are very useful in low level image analysis and synthesis, and even for image compression. Please read the text by F&P at Chapter 7 and 9.2 for details on the pyramids. Specifically we have the following requirements:

- Write your C/C++ code (or Matlab if you still do not know C/C++) to compute the Gaussian and Laplacian Pyramids with 4 or 5 levels depending on the image size. If you use C/C++ programming, you may get 5 extra points.
- You may use either Unix or Windows system, the lab computer or your own notebook.
- You can use PGM image format as the input. See the following link for details on PGM image, http://netpbm.sourceforge.net/doc/pgm.html, or use OpenCV to load images with popular formats.
- You can use XnView (for Windows) or XV (for Unix) to display images or transform images into different formats, such as from *.pgm to *.jpg.
- Some examples images can be accessed at http://www.cs.wisc.edu/~gdguo/courses/examples/Pyramid/ or you can use your own images.
- You need to submit a short report which includes the algorithms you programmed, your code, and the input and output images you obtained.

Report:

Algorithm: Gaussian pyramid and Laplacian pyramid.

A Gaussian pyramid is a collection of images at successively reduced resolutions, generated by applying Gaussian smoothing and downsampling the original image repeatedly.

Steps to Construct a Gaussian Pyramid:

- Start with the original image at the base level.
- Gaussian smoothing: Apply a Gaussian filter to blur the image slightly, reducing high-frequency details.
- Downsampling: Reduce the resolution of the image by subsampling (e.g., keeping every second pixel in each dimension).
- Repeat: Continue smoothing and downsampling to create the next levels.

The resulting pyramid has:

- Level 0 (original resolution) at the bottom.
- Coarser representations (lower resolution) as you move up.

Laplacian Pyramid

A Laplacian pyramid is derived from the Gaussian pyramid and represents the band-pass filtered version of an image (i.e., highlighting the differences between consecutive levels of the Gaussian pyramid). It is particularly useful for reconstructing images or compressing them efficiently.

Steps to Construct a Laplacian Pyramid:

1. Start with the Gaussian pyramid.
2. Upsample: Take an image from a higher level in the Gaussian pyramid and upsample it to the resolution of the current level.
3. Difference image: Subtract the upsampled version from the current Gaussian level to obtain the Laplacian level.
4. Repeat: Perform the process for all levels except the top-most one, which is directly taken from the Gaussian pyramid.

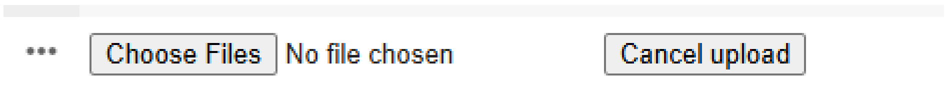
The resulting pyramid has:

- High-frequency details at lower levels.
- Lower frequencies retained as you move up.

Code:

```
!pip install opencv-python-headless
import cv2
import numpy as np
import matplotlib.pyplot as plt
def display_image(img, title="Image"):
    plt.figure(figsize=(6, 6))
    plt.imshow(img, cmap='gray')
    plt.title(title)
    plt.axis('off')
    plt.show()
def display_pyramid(pyramid, title="Pyramid"):
    levels = len(pyramid)
    plt.figure(figsize=(15, 5))
    for i, img in enumerate(pyramid):
        plt.subplot(1, levels, i+1)
        plt.imshow(img, cmap='gray')
        plt.title(f"{title} Level {i}")
        plt.axis('off')
    plt.show()
from google.colab import files
uploaded = files.upload()
image_path = next(iter(uploaded))
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
display_image(image, "Original Image")
levels = 5 if min(image.shape) > 512 else 4
gaussian_pyramid = [image]
for i in range(1, levels):
    downsampled = cv2.pyrDown(gaussian_pyramid[-1])
    gaussian_pyramid.append(downsampled)
display_pyramid(gaussian_pyramid, title="Gaussian Pyramid")
laplacian_pyramid = []
for i in range(levels - 1):
    size = (gaussian_pyramid[i].shape[1], gaussian_pyramid[i].shape[0])  # Width, Height
    upsampled = cv2.pyrUp(gaussian_pyramid[i + 1], dstsize=size)
    laplacian = cv2.subtract(gaussian_pyramid[i], upsampled)
    laplacian_pyramid.append(laplacian)
laplacian_pyramid.append(gaussian_pyramid[-1])  # The smallest Gaussian level is added as the last Laplacian
display_pyramid(laplacian_pyramid, title="Laplacian Pyramid")
```

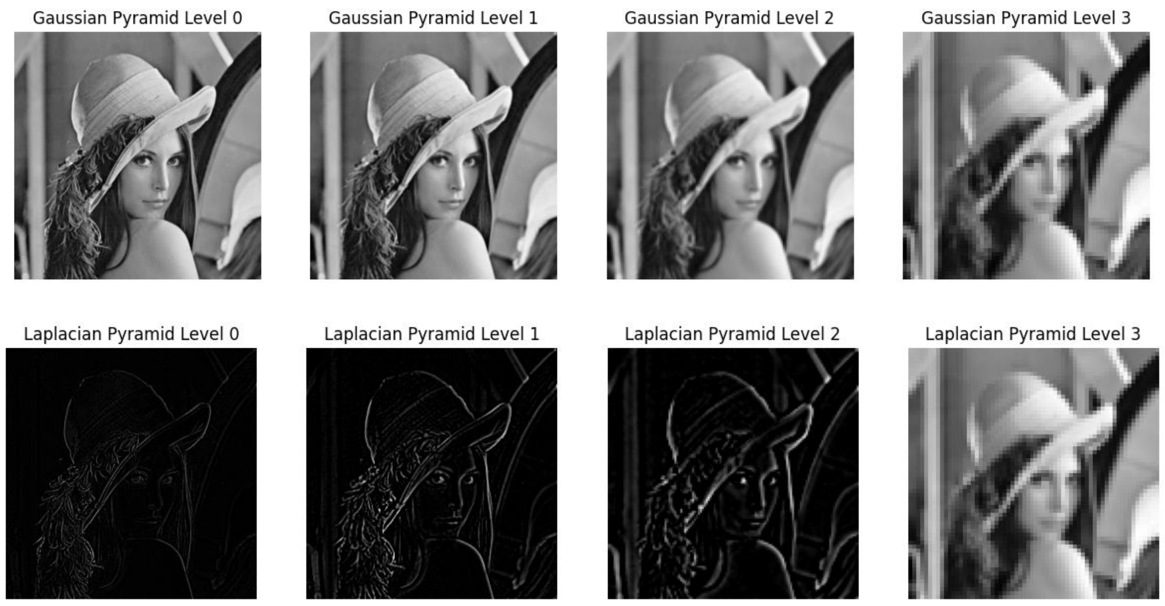
OUTPUT OF THIS CODE



Then Choose the input image file as input then generate the output.

Input Image:

Output Image:


Gaussian Pyramid Level 0
Gaussian Pyramid Level 1
Gaussian Pyramid Level 2
Gaussian Pyramid Level 3
Laplacian Pyramid Level 0
Laplacian Pyramid Level 1
Laplacian Pyramid Level 2
Laplacian Pyramid Level 3