# Image Processing Code Documentation

This document provides an overview of an image processing script with functions to apply convolution, Sobel edge detection,   
Local Binary Pattern (LBP), and Mean-Variance-Median LBP (MVM-LBP) to grayscale images.   
These methods are commonly used for tasks such as feature extraction, edge detection, and texture analysis.

## Modules Required:

- numpy  
- cv2 (OpenCV)  
- matplotlib.pyplot  
- PIL.Image

## Functions:

### 1. convolve(image, kernel)

Performs a 2D convolution on a grayscale image using a given kernel.  
  
Parameters:  
 - image: np.ndarray, the input 2D array representing the grayscale image.  
 - kernel: np.ndarray, the 2D array filter to be applied to the image.  
  
Description:  
 Pads the image to maintain the same output size, applies the kernel over the image,   
 and returns the convolved output of the same size.

### 2. Sobel Edge Detection

Uses two kernels, `sobel\_x` and `sobel\_y`, to detect horizontal and vertical edges, respectively.  
  
Steps:  
 - Applies `sobel\_x` and `sobel\_y` filters using convolve() on the image.  
 - Combines results to compute overall edge intensity (gradient magnitude) of the image.

### 3. compute\_lbp(image)

Calculates the Local Binary Pattern (LBP) for each pixel by comparing it to neighboring pixels.  
  
Parameters:  
 - image: np.ndarray, the grayscale input image.  
  
Description:  
 For each pixel, compares intensity with neighbors, converts the binary pattern to decimal,   
 and assigns it as the LBP value for each pixel. Returns the LBP image.

### 4. mvm\_lbp(image)

Computes the Mean-Variance-Median LBP (MVM-LBP) for texture analysis.  
  
Parameters:  
 - image: np.ndarray, the grayscale input image.  
  
Description:  
 Calculates a threshold based on mean, variance, and median of a 3x3 window.   
 Creates a binary pattern by comparing each pixel to the threshold, converts this pattern to decimal,   
 and assigns it to the output.

## Example Usage:

1. Update `image\_path` with the path to your image.  
2. Load the image in grayscale and apply the functions for feature extraction.  
3. Display the results using matplotlib.