Department of Computer Science and Engineering (Data Science)

Image Processing and Computer Vision I (DJ19DSL603)

Name: Meet Pandya SAPID:60009210202 D22

AIM: To implement frequency domain filters on an image

THEORY:

1. Ideal Low Pass Filter

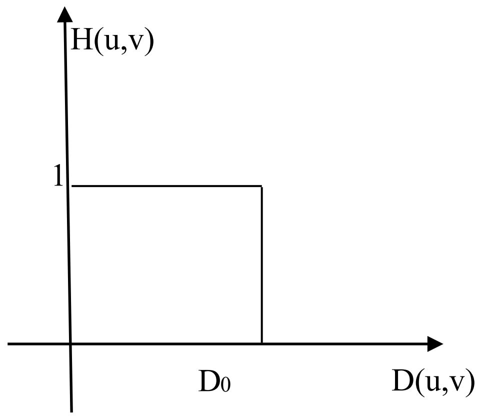
This filter cuts off all high frequency components of the Fourier transform that are at a distance greater than a specified distance D_0.

$$H(u,v) = 1;\ \text{if } D(u,v) < D_0$$

$$= 0;\ \text{if } D(u,v) > D_0$$

Where,

D_0 is the specified non negative distance.



Response of Ideal Low Pass Filter

D(u,v) is the distance from the point (u,v) to the origin of the frequency rectangle for an M X N image.

$$D(u,v)=[(u-M/2)^2 + (v-N/2)^2]^{\wedge}\ ½$$

Therefore ,

For an image, when u=M/2 , v=N/2

$$D(u,v)=0$$

This formula centers our H(u,v).
D(u,v) gives us concentric rings with each ring having a fixed value.

When an ideal low-pass filter is applied to an image, the high-frequency components (i.e., the high-frequency information, such as edges and details) are removed, and only the low-frequency components (i.e., the smooth areas and large details) are retained. This results in a blurring or smoothing effect on the image.

Observations:

1. The image appears smoother or less sharp, as high-frequency details are removed.
2. Edges and other high-contrast features may appear blurred or softened.
3. Noise and other high-frequency artifacts may be reduced, resulting in a cleaner appearance.
4. The overall contrast of the image may be reduced, especially in areas with fine details.
5. The filter may introduce ringing artifacts around edges or high-contrast areas, due to the ideal filter’s inherent characteristics.

2. Ideal High Pass Filter

When an ideal high-pass filter is applied to an image, the low-frequency components (i.e., the smooth areas and large details) are removed, and only the high-frequency components (i.e., the edges and fine details) are retained. This results in an image with enhanced edges and details, but with reduced low-frequency content.

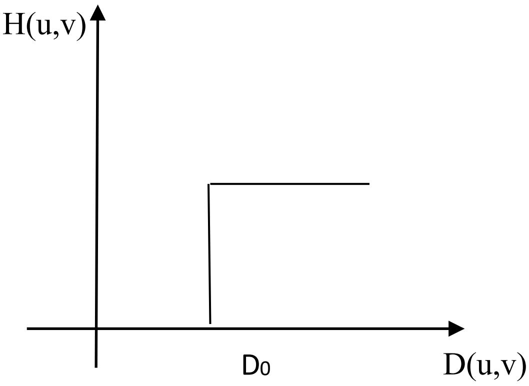
Observations:

1. The image appears sharper, as high-frequency details are enhanced.
2. Edges and other high-contrast features appear more prominent and welldefined.
3. The overall contrast of the image may be increased, especially in areas with fine details.
4. Low-frequency content, such as smooth areas or large features, may appear blurred or reduced in prominence.

The filter may introduce ringing artifacts around edges or high-contrast areas, due to the ideal filter’s inherent characteristics.

Department of Computer Science and Engineering (Data Science)
Image Processing and Computer Vision I (DJ19DSL603)

This filter cuts off all high frequency components of the Fourier transform that are at a distance greater than a specified distance D_0.



Where, H(u,v) = 0; if D(u,v)< D0

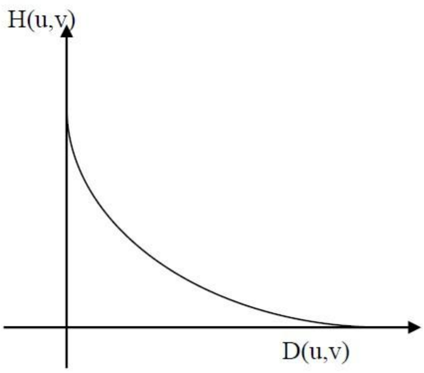
= 1; if D(u,v)> D0

D0 is the specified non negative distance.

D(u,v) is the distance from the point (u,v) to the origin of the frequency rectangle for an M X N image.

3. Gaussian Low Pass Filter Gaussian LPF is given by:

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2}$$



Where, σ is the standard deviation and is a measure of spread of the Gaussian curve. If we put σ =D0 we get, $H(u,v) = e^{-D^2(u,v)/2D0^2}$

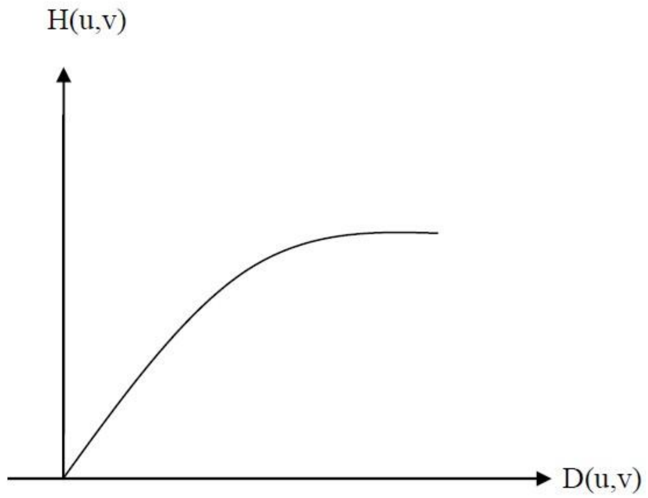
The response of the Gaussian LPF is similar to that of BLPF but there are no ringing effects.

4. Gaussian High Pass Filter

The basic formula is, $H_{hp}(u,v)$
$= 1 - H_{lp}(u,v)$

Therefore,

$H_{Gaussian\ hp}(u,v) = 1 - H_{Gaussian\ lp}(u,v)$
$H_{GHPF} = 1 - e^{-D^2(u,v)/2D0^2}$



The results of Gaussian high pass filter are smoother and cleaner

Department of Computer Science and Engineering (Data Science)
Image Processing and Computer Vision I (DJ19DSL603)

Lab Assignments to complete in this session

Problem Statement: Develop a Python program utilizing the OpenCV library to manipulate images from the Fashion MNIST digits dataset. The program should address the following tasks:

1. Importing libraries
2. Read random image(s) from the MNIST fashion dataset.
3. **Dataset Link:** Fashion MNIST Github
4. Getting the Fourier Transform
5. Ideal Low Pass Filtering
6. Multiplication between the Fourier Transformed input image and the filtering mask
7. Taking Inverse Fourier Transform of the convoluted image
8. Ideal High Pass Filtering
9. Multiplication between the Fourier Transformed input image and the filtering mask
10. Taking Inverse Fourier Transform of the convoluted image
11. Gaussian Low Pass Filtering
12. Multiplication between the Fourier Transformed input image and the filtering mask
13. Taking Inverse Fourier Transform of the convoluted image
14. Gaussian High Pass Filtering
15. Multiplication between the Fourier Transformed input image and the filtering mask
16. Taking Inverse Fourier Transform of the convoluted image

The solution to the operations performed must be produced by scratch coding without the use of built in OpenCV methods.

CODE:

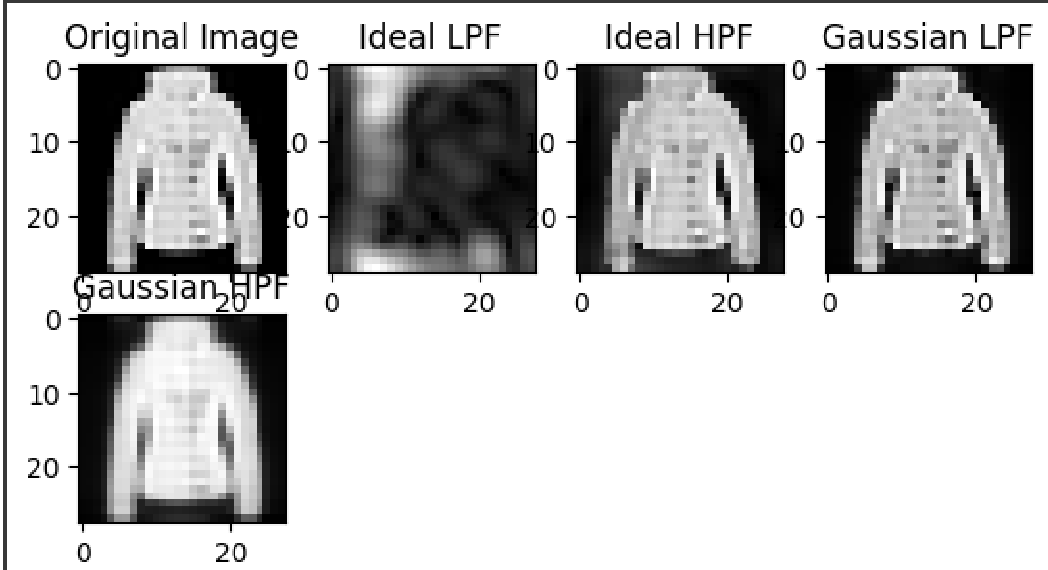
```
import pandas as pd
import numpy as np
from google.colab.patches import cv2_imshow
import cv2
!pip install mnist
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import fashion_mnist
from scipy.fft import fft2, ifft2

def read_random_image():
    (images, _), _ = fashion_mnist.load_data() # Fixed typo, changed "C_)" to
"(images, _)"
    idx = np.random.randint(len(images))
    return images[idx]

# Function to perform Fourier transform
def fourier_transform(img):
    return fft2(img)

# Function to apply ideal low pass filtering
def ideal_low_pass_filtering(fourier_img, cutoff_freq):
    rows, cols = fourier_img.shape
    crow, ccol = rows // 2, cols // 2
    mask = np.zeros((rows, cols))
    mask[crow - cutoff_freq:crow + cutoff_freq, ccol - cutoff_freq:ccol +
cutoff_freq] = 1 # Fixed typo, changed "cutoff_freql" to "cutoff_freq"
    return fourier_img * mask

# Function to apply ideal high pass filtering
def ideal_high_pass_filtering(fourier_img, cutoff_freq):
    return fourier_img - ideal_low_pass_filtering(fourier_img, cutoff_freq)

# Function to apply gaussian low pass filtering
def gaussian_low_pass_filtering(fourier_img, sigma):
    rows, cols = fourier_img.shape
    crow, ccol = rows // 2, cols // 2
    x = np.arange(cols) - ccol
    y = np.arange(rows) - crow
    X, Y = np.meshgrid(x, y)
    mask = np.exp(-(X ** 2 + Y ** 2) / (2 * sigma ** 2))
    return fourier_img * mask
```

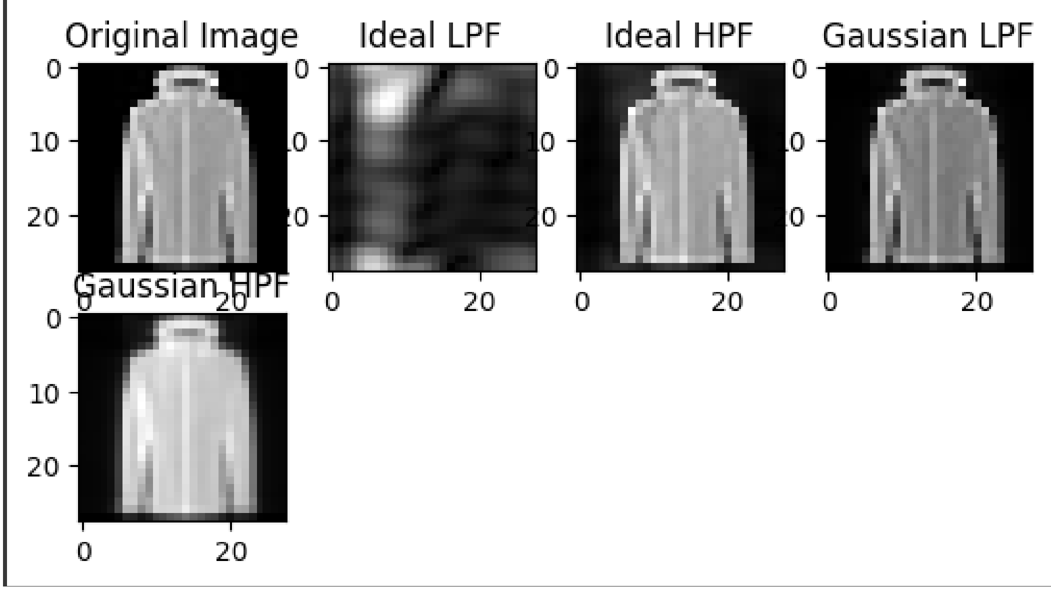
```
# Function to apply gaussian high pass filtering
def gaussian_high_pass_filtering(fourier_img, sigma):
    return fourier_img - gaussian_low_pass_filtering(fourier_img, sigma)

# Function to perform inverse Fourier transform
def inverse_fourier_transform(fourier_img):
    return np.abs(ifft2(fourier_img))

def main():
    img = read_random_image()
    plt.figure()
    plt.subplot(3, 4, 1)
    plt.title("Original Image")
    plt.imshow(img, cmap='gray')
    fourier_img = fourier_transform(img)
    cutoff_freq = 20
    low_pass_img =
inverse_fourier_transform(ideal_low_pass_filtering(fourier_img, cutoff_freq))
    high_pass_img =
inverse_fourier_transform(ideal_high_pass_filtering(fourier_img,
cutoff_freq))
    sigma = 20
    gaussian_low_pass_img =
inverse_fourier_transform(gaussian_low_pass_filtering(fourier_img, sigma))
    gaussian_high_pass_img =
inverse_fourier_transform(gaussian_high_pass_filtering(fourier_img, sigma))
    plt.subplot(3, 4, 2)
    plt.title("Ideal LPF")
    plt.imshow(low_pass_img, cmap='gray')
    plt.subplot(3, 4, 3)
    plt.title("Ideal HPF")
    plt.imshow(high_pass_img, cmap='gray')
    plt.subplot(3, 4, 4)
    plt.title("Gaussian LPF")
    plt.imshow(gaussian_low_pass_img, cmap='gray')
    plt.subplot(3, 4, 5)
    plt.title("Gaussian HPF")
    plt.imshow(gaussian_high_pass_img, cmap='gray')
    plt.show()

if __name__ == "__main__":
    for i in range(10):
      main()
```

Department of Computer Science and Engineering (Data Science)

Image Processing and Computer Vision I (DJ19DSL603)

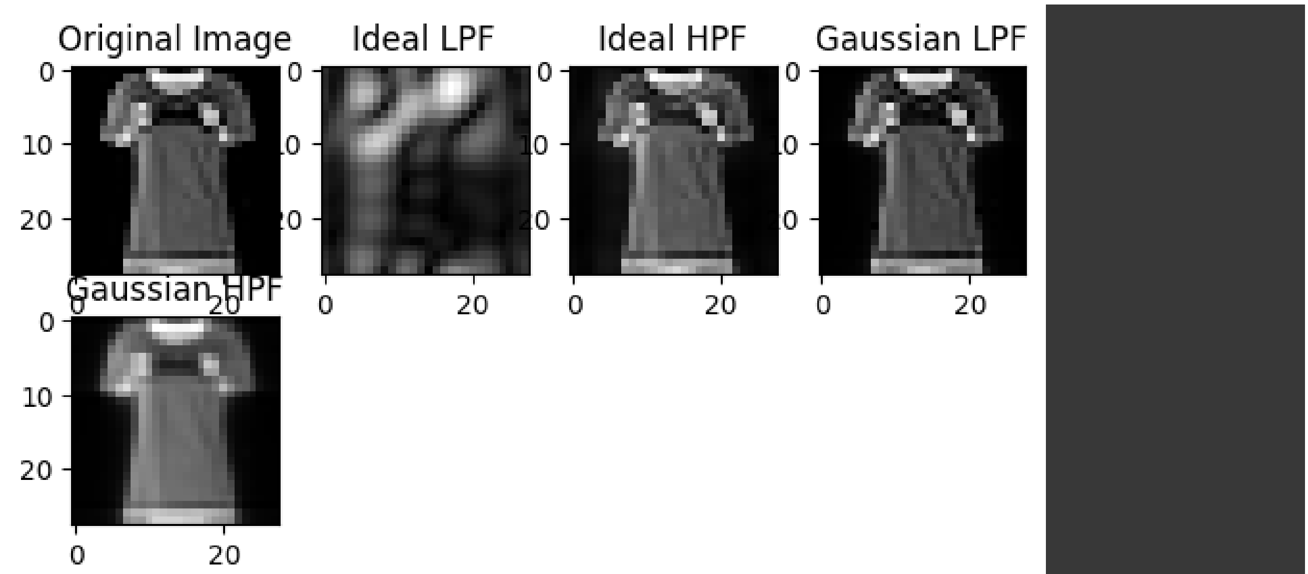
```
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from mnist) (1.25.2)
```

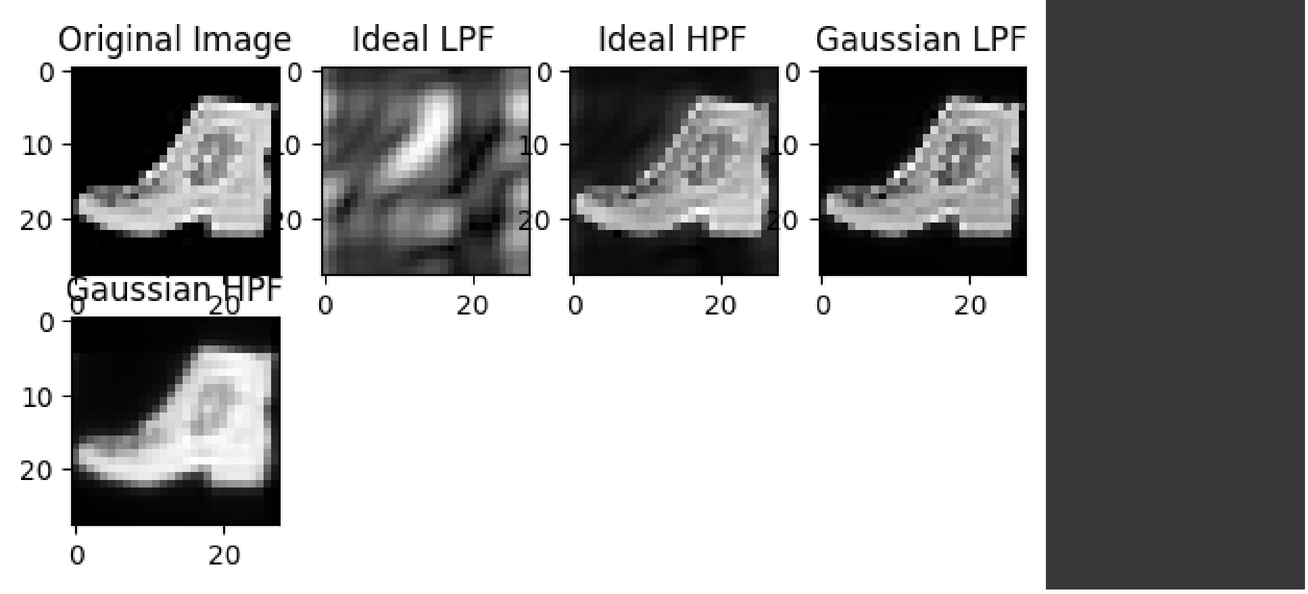


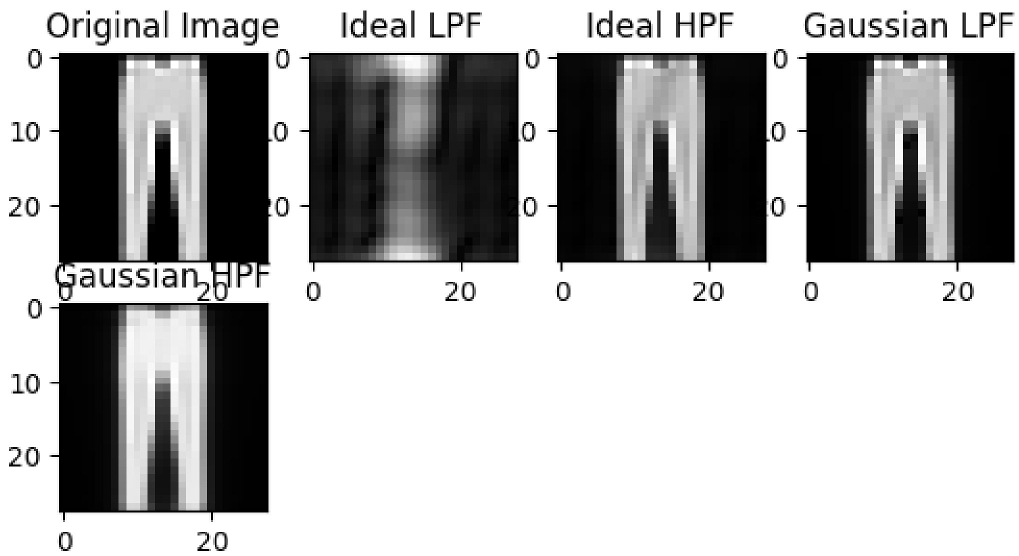


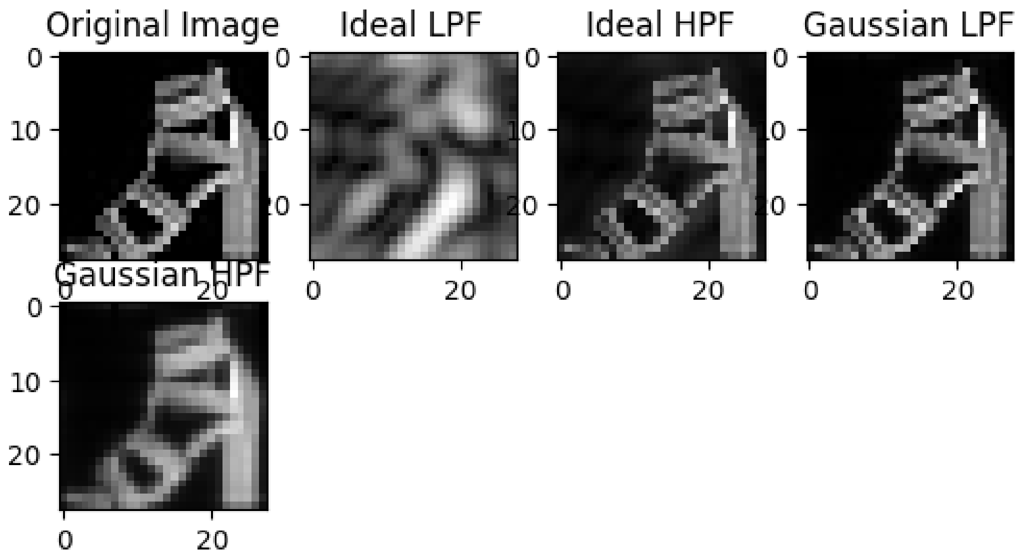
Department of Computer Science and Engineering (Data Science)

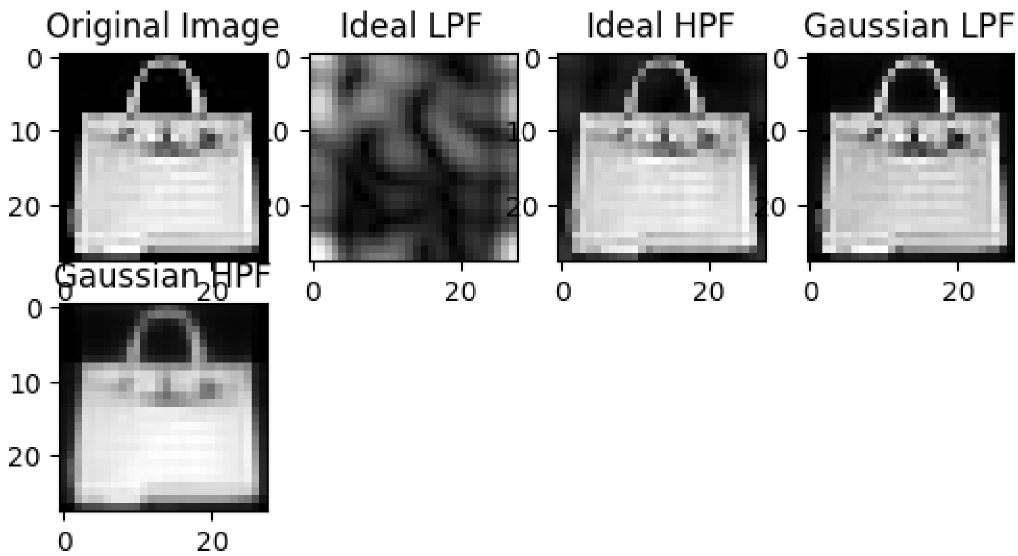
Image Processing and Computer Vision I (DJ19DSL603)

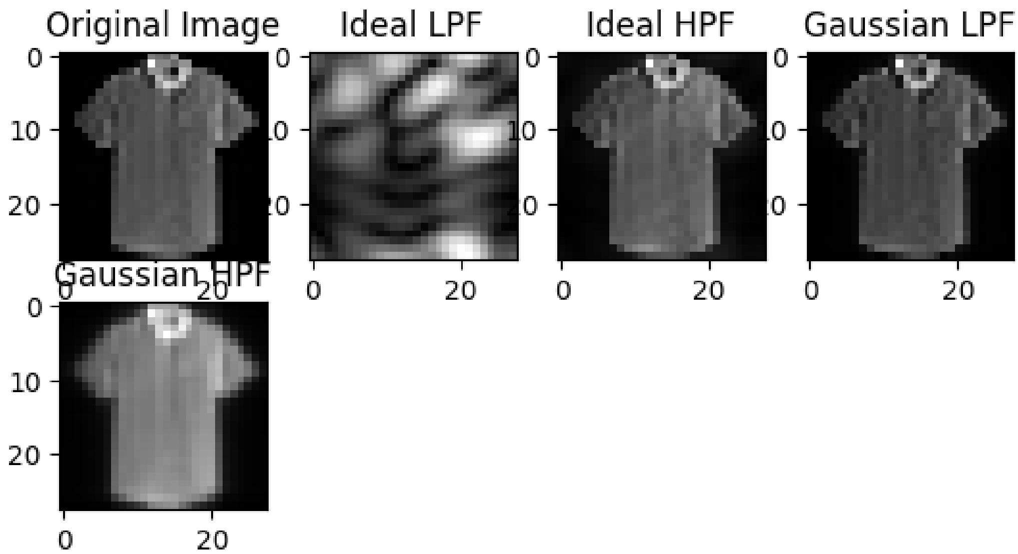


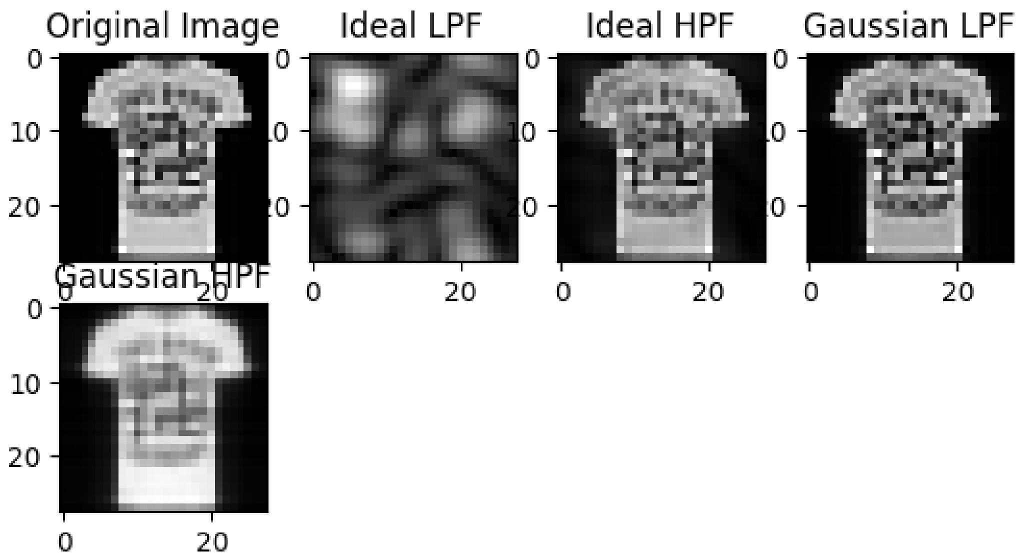


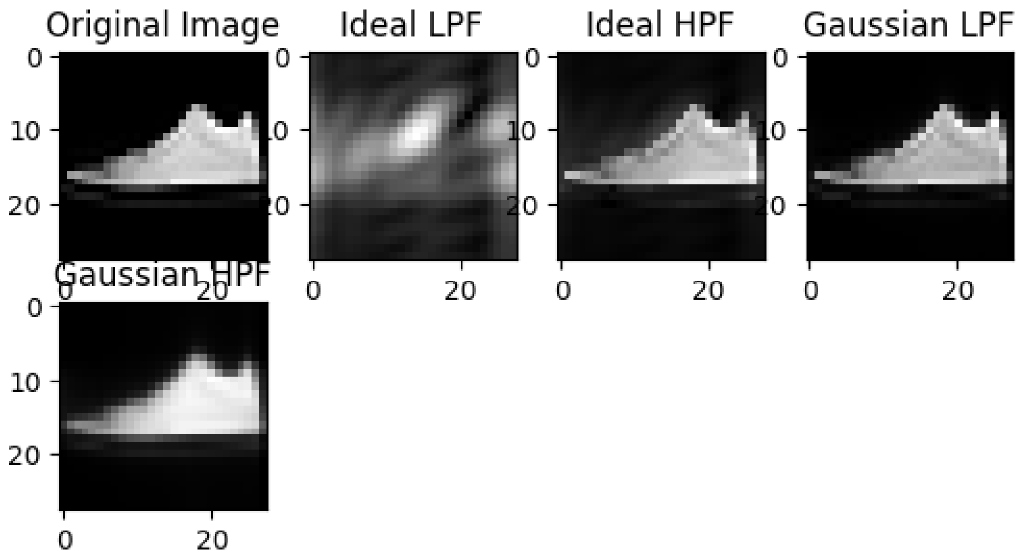

Original Image
Ideal LPF
Ideal HPF
Gaussian LPF
Gaussian HPF


Original Image
Ideal LPF
Ideal HPF
Gaussian LPF
Gaussian HPF


Original Image
Ideal LPF
Ideal HPF
Gaussian LPF
Gaussian HPF


Original Image
Ideal LPF
Ideal HPF
Gaussian LPF
Gaussian HPF


Original Image
Ideal LPF
Ideal HPF
Gaussian LPF
Gaussian HPF


Original Image
Ideal LPF
Ideal HPF
Gaussian LPF
Gaussian HPF