

231501037

EXP NO: 04

DATE: 25-07-2025

Low pass and High pass filtering mechanisms

Aim: To Implement the various low pass and high pass filtering mechanisms.

Algorithm:

1. Read and convert image to grayscale.
2. Apply low pass (Gaussian/average) filter using cv2.GaussianBlur() or cv2.blur().
3. Apply high pass filter using Laplacian or Sobel operator.
4. Combine results to observe smoothing vs sharpening effects.
5. Display filtered images.
6. Save outputs.

Code:

```
import cv2

import numpy as np

from PIL import Image

import matplotlib.pyplot as plt

# Read image using PIL and convert to OpenCV format
try:

    img_pil = Image.open('/content/drive/MyDrive/input.jpg')

    img = cv2.cvtColor(np.array(img_pil), cv2.COLOR_RGB2BGR)

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

except Exception as e:

    print(f"Error reading image: {e}")

    img = None

    gray = None

if gray is not None:
```

231501037

--- Low Pass Filters ---

avg_blur = cv2.blur(gray, (5,5)) # Average filter

gauss_blur = cv2.GaussianBlur(gray, (5,5), 0) # Gaussian filter

median_blur = cv2.medianBlur(gray, 5) # Median filter

--- High Pass Filters ---

laplacian = cv2.Laplacian(gray, cv2.CV_64F)

laplacian = cv2.convertScaleAbs(laplacian)

sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=3)

sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=3)

sobel = cv2.convertScaleAbs(cv2.magnitude(sobelx, sobely))

High-pass kernel filter

kernel = np.array([[[-1,-1,-1],

 [-1, 8,-1],

 [-1,-1,-1]])

highpass = cv2.filter2D(gray, -1, kernel)

Display results using matplotlib

images = {

 "Original": gray,

 "Average Blur": avg_blur,

 "Gaussian Blur": gauss_blur,

 "Median Blur": median_blur,

 "Laplacian": laplacian,

 "Sobel": sobel,

 "High-pass Kernel": highpass

231501037

```
}
```

```
plt.figure(figsize=(15, 10))
```

```
for i, (title, img) in enumerate(images.items()):
```

```
    plt.subplot(2, 4, i + 1)
```

```
    plt.imshow(img, cmap='gray')
```

```
    plt.title(title)
```

```
    plt.axis('off')
```

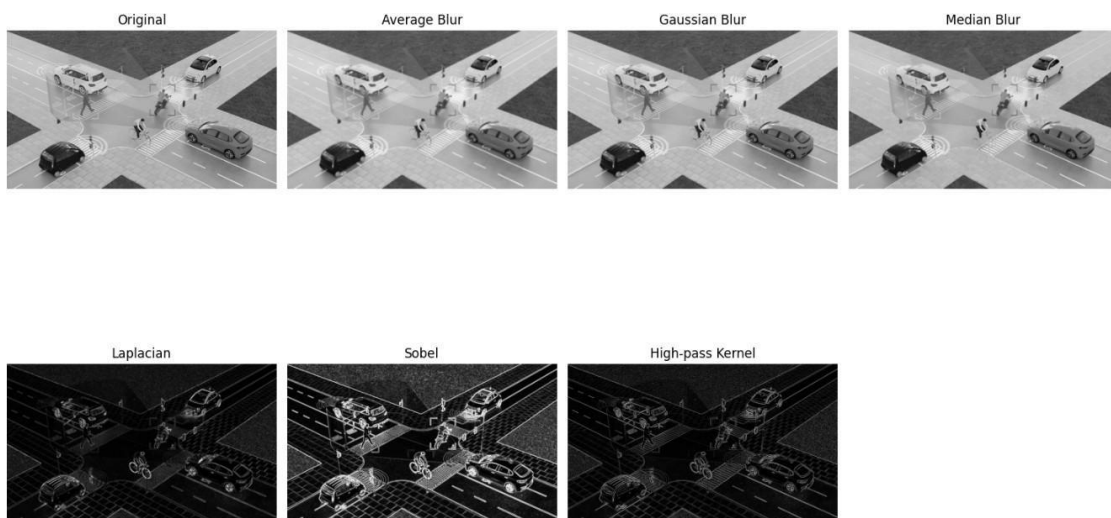
```
plt.tight_layout()
```

```
plt.show()
```

else:

```
    print("Could not process the image.")
```

Output:



231501037

Code:

```
from google.colab.patches import cv2_imshow

import cv2

import numpy as np

from PIL import Image

# Read image using PIL and convert to OpenCV format

try:

    img_pil = Image.open('/content/drive/MyDrive/input.jpg') # Replace with an existing
image file

    img = cv2.cvtColor(np.array(img_pil), cv2.COLOR_RGB2BGR)

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

except Exception as e:

    print(f"Error reading image: {e}")

    img = None

    gray = None

if gray is not None:

    # --- Low Pass Filters ---

    avg_blur = cv2.blur(gray, (5,5))          # Average filter

    gauss_blur = cv2.GaussianBlur(gray, (5,5), 0) # Gaussian filter

    median_blur = cv2.medianBlur(gray, 5)      # Median filter


    # --- High Pass Filters ---

    laplacian = cv2.Laplacian(gray, cv2.CV_64F)

    laplacian = cv2.convertScaleAbs(laplacian)


    sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=3)

    sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=3)
```

231501037

```
sobel = cv2.convertScaleAbs(cv2.magnitude(sobelx, sobely))
```

```
# High-pass kernel filter
```

```
kernel = np.array([[[-1,-1,-1],  
                    [-1, 8,-1],  
                    [-1,-1,-1]])
```

```
highpass = cv2.filter2D(gray, -1, kernel)
```

```
# Display results using cv2_imshow
```

```
cv2_imshow(gray)
```

```
cv2_imshow(avg_blur)
```

```
cv2_imshow(gauss_blur)
```

```
cv2_imshow(median_blur)
```

```
cv2_imshow(laplacian)
```

```
cv2_imshow(sobel)
```

```
cv2_imshow(highpass)
```

```
# The waitKey and destroyAllWindows are not needed when using cv2_imshow
```

```
# cv2.waitKey(0)
```

```
# cv2.destroyAllWindows()
```

```
# Save results
```

```
cv2.imwrite("avg_blur.jpg", avg_blur)
```

```
cv2.imwrite("gauss_blur.jpg", gauss_blur)
```

```
cv2.imwrite("median_blur.jpg", median_blur)
```

```
cv2.imwrite("laplacian.jpg", laplacian)
```

```
cv2.imwrite("sobel.jpg", sobel)
```

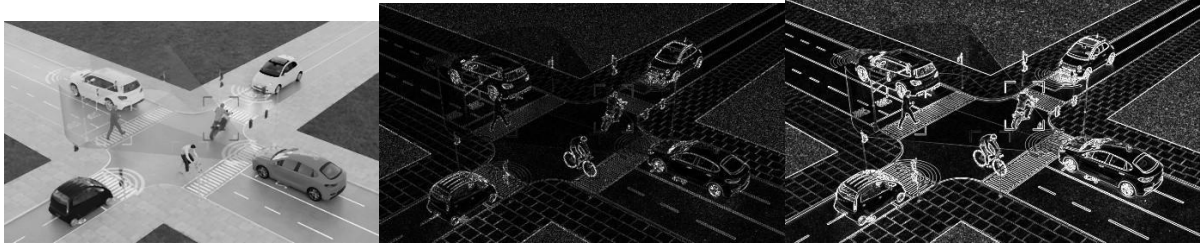
```
cv2.imwrite("highpass.jpg", highpass)
```

231501037

else:

```
print("Could not process the image.")
```

Output:



Result: Thus, the various low pass and high pass filtering mechanisms was successfully implemented.