MACHINE VISION

LAB 4: Median filter, Min filter, Max filter

NAME : OM SUBRATO DEY

REGISTER NUMBER : 21BAI1876

1. Median Filter:

CODE:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image
image_path = 'OMPHOTO.jpeg'  # Replace with your image path
image = cv2.imread(image_path, cv2.IMREAD_COLOR)

# Convert image to RGB (OpenCV loads images in BGR by default)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Apply median filter
# kernel_size should be an odd integer (e.g., 3, 5, 7)
kernel_size = 5
filtered_image = cv2.medianBlur(image_rgb, kernel_size)

# Display the original and filtered images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image_rgb)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title('Filtered Image')
plt.imshow(filtered_image)
```

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

OUTPUT:

jupyter 21BAI1876_LAB_4 Last Checkpoint: 11 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [1]:
```
#MEDIAN FILTER
```

In [2]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
# Load the image
image_path = 'OMPHOTO.jpeg'  # Replace with your image path
image = cv2.imread(image_path, cv2.IMREAD_COLOR)
```

In [4]:
```
# Convert image to RGB (OpenCV loads images in BGR by default)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

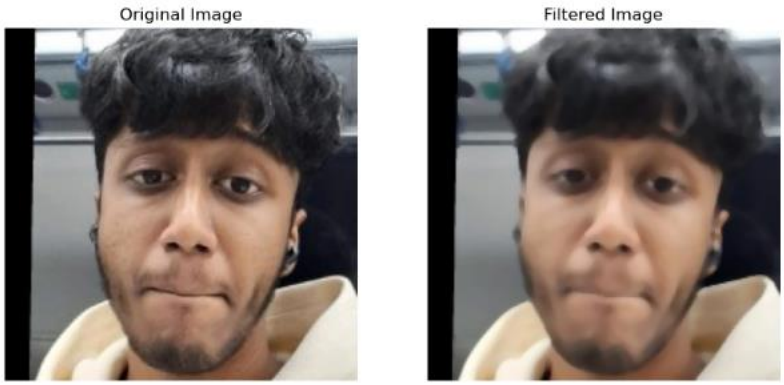
In [5]:
```
# Apply median filter
# kernel_size should be an odd integer (e.g., 3, 5, 7)
kernel_size = 5
filtered_image = cv2.medianBlur(image_rgb, kernel_size)
```

In [6]:
```
# Display the original and filtered images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image_rgb)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title('Filtered Image')
plt.imshow(filtered_image)
plt.axis('off')

plt.show()
```



2. Min Filter:

CODE:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


def min_filter(image, kernel_size):
    # Pad the image to handle border effects
    pad_size = kernel_size // 2
    padded_image = np.pad(image, ((pad_size, pad_size), (pad_size, pad_size), (0, 0)), mode='edge')


    # Output image
    filtered_image = np.zeros_like(image)


    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            # Extract the neighborhood
            neighborhood = padded_image[i:i+kernel_size, j:j+kernel_size]
            # Apply min filter
            filtered_image[i, j] = np.min(neighborhood, axis=(0, 1))


    return filtered_image


# Load the image
image_path = 'OMPHOTO.jpeg'  # Replace with your image path
image = cv2.imread(image_path, cv2.IMREAD_COLOR)
```

```
# Convert image to RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Apply min filter
kernel_size = 5
min_filtered_image = min_filter(image_rgb, kernel_size)

# Display the original and min filtered images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image_rgb)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title('Min Filtered Image')
plt.imshow(min_filtered_image)
plt.axis('off')

plt.show()
```

OUTPUT:

jupyter 21BAI1876_LAB_4 Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run

In [7]:
```
# MIN FILTER
```

In [8]:
```
def min_filter(image, kernel_size):
    # Pad the image to handle border effects
    pad_size = kernel_size // 2
    padded_image = np.pad(image, ((pad_size, pad_size), (pad_size, pad_size), (0, 0)), mode='edge')

    # Output image
    filtered_image = np.zeros_like(image)

    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            # Extract the neighborhood
            neighborhood = padded_image[i:i+kernel_size, j:j+kernel_size]
            # Apply min filter
            filtered_image[i, j] = np.min(neighborhood, axis=(0, 1))

    return filtered_image
```

In [9]:
```
# Load the image
image_path = 'OMPHOTO.jpeg'  # Replace with your image path
image = cv2.imread(image_path, cv2.IMREAD_COLOR)

# Convert image to RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

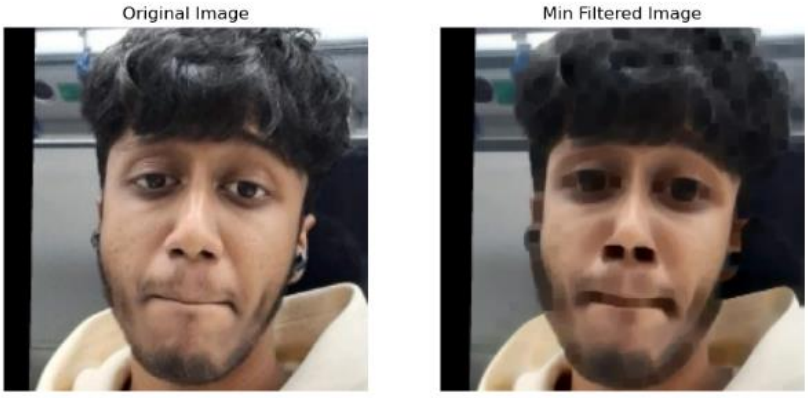
In [10]:
```
# Apply min filter
kernel_size = 5
min_filtered_image = min_filter(image_rgb, kernel_size)
```

In [11]:
```
# Display the original and min filtered images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image_rgb)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title('Min Filtered Image')
plt.imshow(min_filtered_image)
plt.axis('off')

plt.show()
```



3. Max Filter

CODE:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


def max_filter(image, kernel_size):
    # Pad the image to handle border effects
    pad_size = kernel_size // 2
    padded_image = np.pad(image, ((pad_size, pad_size), (pad_size, pad_size), (0, 0)), mode='edge')


    # Output image
    filtered_image = np.zeros_like(image)


    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            # Extract the neighborhood
            neighborhood = padded_image[i:i+kernel_size, j:j+kernel_size]
            # Apply max filter
            filtered_image[i, j] = np.max(neighborhood, axis=(0, 1))


    return filtered_image


# Load the image
image_path = 'OMPHOTO.jpeg'  # Replace with your image path
image = cv2.imread(image_path, cv2.IMREAD_COLOR)
```

```
# Convert image to RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Apply max filter
kernel_size = 5
max_filtered_image = max_filter(image_rgb, kernel_size)

# Display the original and max filtered images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image_rgb)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title('Max Filtered Image')
plt.imshow(max_filtered_image)
plt.axis('off')

plt.show()
```

OUTPUT:

jupyter 21BAI1876_LAB_4 Last Checkpoint: a minute ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

```
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            # Extract the neighborhood
            neighborhood = padded_image[i:i+kernel_size, j:j+kernel_size]
            # Apply max filter
            filtered_image[i, j] = np.max(neighborhood, axis=(0, 1))

    return filtered_image
```

In [15]:
```
# Load the image
image_path = 'OMPHOTO.jpeg'  # Replace with your image path
image = cv2.imread(image_path, cv2.IMREAD_COLOR)

# Convert image to RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

In [16]:
```
# Apply max filter
kernel_size = 5
max_filtered_image = max_filter(image_rgb, kernel_size)
```

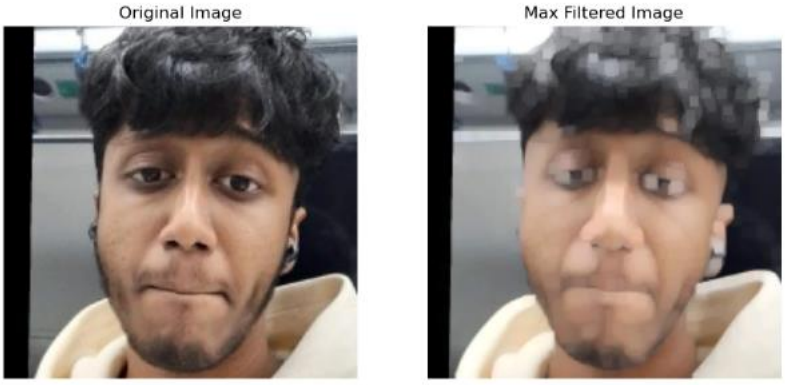
In [17]:
```
# Apply max filter
kernel_size = 5
max_filtered_image = max_filter(image_rgb, kernel_size)

# Display the original and max filtered images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image_rgb)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title('Max Filtered Image')
plt.imshow(max_filtered_image)
plt.axis('off')

plt.show()
```



In [ ]: