Lab Assignment 2

Problem Statement

Write a Python program to simulate different **Linear** and **Non-Linear filters** on a given image. The program should read an input image and apply various types of filters to enhance, smooth, or process the image. Finally, visualize the results for comparison.

Hints and Guidelines

Tools to Use:

1. **OpenCV**: For reading images, applying filters, and image processing.
2. **NumPy**: For creating kernels and handling numerical computations.
3. **Matplotlib**: For visualizing the original and filtered images.

Steps to Approach the Problem:

1. **Load the Image**:
    - Use `cv2.imread` to read the image in BGR format.
    - Convert the image to RGB format for better visualization with Matplotlib using `cv2.cvtColor`.
2. **Implement Filters**:
    - Define functions to apply **linear** and **non-linear filters**.
    - **Linear Filters**:
        - Create a kernel (e.g., a 5x5 averaging filter) using NumPy.
        - Use `cv2.filter2D` to apply the kernel.
    - **Non-Linear Filters**:
        - Use built-in OpenCV functions like:
            - `cv2.GaussianBlur` for Gaussian filtering.
            - `cv2.medianBlur` for Median filtering.
            - `cv2.bilateralFilter` for Bilateral filtering.
            - `cv2.erode` and `cv2.dilate` for Min and Max filtering.
3. **Visualize Results**:
    - Use Matplotlib's `plt.subplots` to create a grid for displaying images side by side.
    - Use `axs[i].imshow` to display each filtered image with a title indicating the filter applied.
4. **Compare and Analyze**:
    - Observe how different filters affect the input image (e.g., smoothness, sharpness, noise removal).

Pseudo Code:

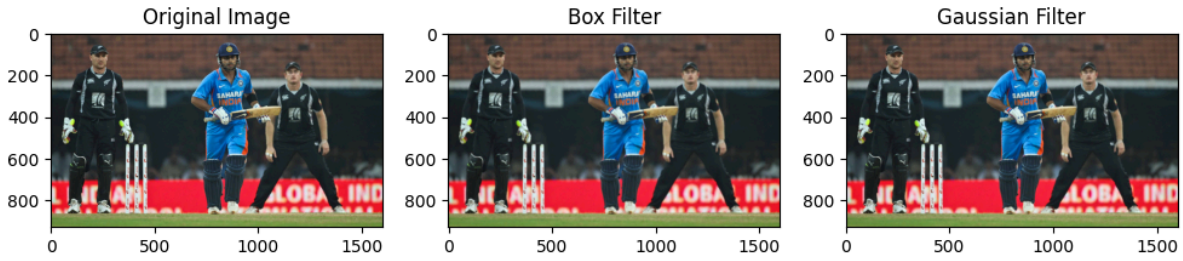
```
plaintext
1. Import required libraries (cv2, numpy, matplotlib).
2. Define a function to apply linear filters:
   - Use `cv2.filter2D` with a custom kernel.
3. Define a function to apply non-linear filters:
   - Use OpenCV functions for median, Gaussian, bilateral, min, and
max filters.
4. Read an image using `cv2.imread`.
5. Display the original image.
6. Apply the linear and non-linear filters using the defined
functions.
7. Visualize the original and filtered images using Matplotlib.
```

In [1]:
```
# Download assignment files
!wget https://github.com/buntyke/vnr_dlcv2024_labs/releases/download/DLCVLab2/virat
```

```
--2024-12-08 06:56:08--  https://github.com/buntyke/vnr_dlcv2024_labs/releases/dow
nload/DLCVLab2/virat-kohli.jpg
Resolving github.com (github.com)... 20.27.177.113
Connecting to github.com (github.com)|20.27.177.113|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e
65be/878811324/f8e3cc53-6cc8-446d-b1ce-2e161a17cd76?X-Amz-Algorithm=AWS4-HMAC-SHA2
56&X-Amz-Credential=releaseassetproduction%2F20241208%2Fus-east-1%2Fs3%2Faws4_requ
est&X-Amz-Date=20241208T065609Z&X-Amz-Expires=300&X-Amz-Signature=3f99bbee0a7369ca
617795f05c3262b83fcc6ee8cd37b8f954957aa218c3299b&X-Amz-SignedHeaders=host&response
-content-disposition=attachment%3B%20filename%3Dvirat-kohli.jpg&response-content-t
ype=application%2Foctet-stream [following]
--2024-12-08 06:56:09--  https://objects.githubusercontent.com/github-production-r
elease-asset-2e65be/878811324/f8e3cc53-6cc8-446d-b1ce-2e161a17cd76?X-Amz-Algorithm
=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20241208%2Fus-east-1%2
Fs3%2Faws4_request&X-Amz-Date=20241208T065609Z&X-Amz-Expires=300&X-Amz-Signature=3
f99bbee0a7369ca617795f05c3262b83fcc6ee8cd37b8f954957aa218c3299b&X-Amz-SignedHeader
s=host&response-content-disposition=attachment%3B%20filename%3Dvirat-kohli.jpg&res
ponse-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.19
9.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 243387 (238K) [application/octet-stream]
Saving to: ‘virat-kohli.jpg’

virat-kohli.jpg     100%[===================>] 237.68K  1.42MB/s    in 0.2s

2024-12-08 06:56:10 (1.42 MB/s) - ‘virat-kohli.jpg’ saved [243387/243387]
```

In [2]:
```
### WRITE CODE HERE ###
import cv2
import numpy as np
from matplotlib import pyplot as plt

def apply_linear_filter(image, kernel):
    filtered_image = cv2.filter2D(image, -1, kernel)
    return filtered_image

def apply_nonlinear_filter(image, filter_type):
    if filter_type == "median":
        filtered_image = cv2.medianBlur(image, 5)
```

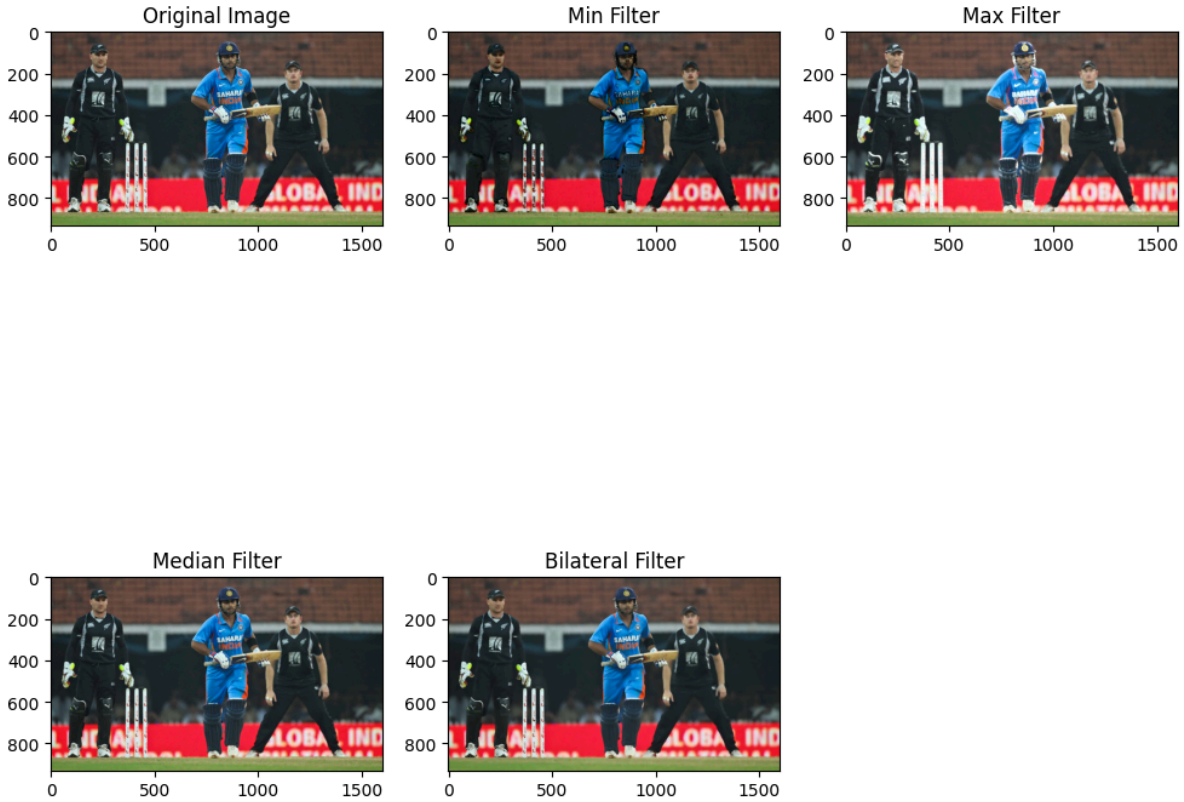
```
    elif filter_type == "gaussian":
        filtered_image = cv2.GaussianBlur(image, (5, 5), 0)
    elif filter_type == "bilateral":
        filtered_image = cv2.bilateralFilter(image,15,75,75)
    elif filter_type == "min":
        filtered_image = cv2.erode(image,
                cv2.getStructuringElement(cv2.MORPH_RECT, (5,5)))
    elif filter_type == "max":
        filtered_image = cv2.dilate(image,
                cv2.getStructuringElement(cv2.MORPH_RECT, (5,5)))
    else:
        raise ValueError("Invalid non-linear filter type.")
    return filtered_image

image_path = "./virat-kohli.jpg"
image = cv2.imread(image_path)

fig, axs = plt.subplots(1,3,figsize=(10,5))

# Display the original image
axs[0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
axs[0].set_title("Original Image")

# Box Filter 5X5
kernel = np.ones((5, 5), np.float32) / 25
linear_filtered_image = apply_linear_filter(image, kernel)
axs[1].imshow(cv2.cvtColor(linear_filtered_image, cv2.COLOR_BGR2RGB))
axs[1].set_title("Box Filter")

# Gausian Filter
gaussian_filtered_image = apply_nonlinear_filter(image, "gaussian")
axs[2].imshow(cv2.cvtColor(gaussian_filtered_image, cv2.COLOR_BGR2RGB))
axs[2].set_title("Gaussian Filter")

plt.tight_layout()
plt.show()
```



In [3]:

```
fig, axs = plt.subplots(2,3,figsize=(10,10))

# Display the original image
axs[0,0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
axs[0,0].set_title("Original Image")

# Min Filter 5 x 5
min_filtered_image = apply_nonlinear_filter(image, "min")
axs[0,1].imshow(cv2.cvtColor(min_filtered_image, cv2.COLOR_BGR2RGB))
axs[0,1].set_title("Min Filter")

# Max Filter 5 x 5
max_filtered_image = apply_nonlinear_filter(image, "max")
axs[0,2].imshow(cv2.cvtColor(max_filtered_image, cv2.COLOR_BGR2RGB))
axs[0,2].set_title("Max Filter")

# Medain Filter 5 x 5
median_filtered_image = apply_nonlinear_filter(image, "median")
```

```
axs[1,0].imshow(cv2.cvtColor(median_filtered_image, cv2.COLOR_BGR2RGB))
axs[1,0].set_title("Median Filter")

# Biltaeral Filter
bil_filtered_image = apply_nonlinear_filter(image, "bilateral")
axs[1,1].imshow(cv2.cvtColor(bil_filtered_image, cv2.COLOR_BGR2RGB))
axs[1,1].set_title("Bilateral Filter")

axs[1,2].axis('off')
plt.tight_layout()
plt.show()
```



In [ ]: