NARESHDC / Implementation-of-filter

Code Pull requests Actions Projects Wiki Security Insights

main Implementation-of-filter / README.md

NARESHDC Update README.md adc7b30 · now

195 lines (132 loc) · 3.6 KB

Implementation-of-filter

Aim:

To implement filters for smoothing and sharpening the images in the spatial domain.

Software Required:

Anaconda - Python 3.7

Algorithm:

Step1

Import the required libraries.

Step2

Convert the image from BGR to RGB.

Step3

Apply the required filters for the image separately.

Step4

Plot the original and filtered image by using matplotlib.pyplot.

Step5

End the program.

Program:

```
Developed By   : NARESH M
Register Number: 212223220064
```

1. Smoothing Filters

i) Using Averaging Filter

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
image1=cv2.imread("rome.jpg")
image2=cv2.cvtColor(image1,cv2.COLOR_BGR2RGB)
kernel=np.ones((11,11),np.float32)/169
image3=cv2.filter2D(image2,-1,kernel)
plt.figure(figsize=(9,9))
plt.subplot(1,2,1)
plt.imshow(image2)
plt.title("Original Image")
plt.axis("off")
plt.subplot(1,2,2)
plt.imshow(image3)
plt.title("Average Filter Image")
plt.axis("off")
plt.show()
```

ii) Using Weighted Averaging Filter

```
kernel1=np.array([[1,2,1],[2,4,2],[1,2,1]])/16
image3=cv2.filter2D(image2,-1,kernel1)
plt.figure(figsize=(9,9))
plt.subplot(1,2,1)
plt.imshow(image2)
plt.title("Original Image")
plt.axis("off")
plt.subplot(1,2,2)
plt.imshow(image3)
plt.title("Weighted Average Filter Image")
plt.axis("off")
plt.show()
```

iii) Using Gaussian Filter

```
gaussian_blur=cv2.GaussianBlur(image2,(33,33),0,0)
plt.figure(figsize=(9,9))
plt.subplot(1,2,1)
plt.imshow(image2)
plt.title("Original Image")
plt.axis("off")
plt.subplot(1,2,2)
plt.imshow(gaussian_blur)
plt.title("Gaussian Blur")
plt.axis("off")
plt.show()
```

iv) Using Median Filter

```
median=cv2.medianBlur(image2,13)
plt.figure(figsize=(9,9))
plt.subplot(1,2,1)
plt.imshow(image2)
plt.title("Original Image")
plt.axis("off")
plt.subplot(1,2,2)
plt.imshow(median)
plt.title("Median Blur")
plt.axis("off")
plt.show()
```

2. Sharpening Filters

i) Using Laplacian Kernal

```
kernel2=np.array([[-1,-1,-1],[2,-2,1],[2,1,-1]])
image3=cv2.filter2D(image2,-1,kernel2)
plt.figure(figsize=(9,9))
plt.subplot(1,2,1)
plt.imshow(image2)
plt.title("Original Image")
plt.axis("off")
plt.subplot(1,2,2)
plt.imshow(image3)
plt.title("Laplacian Kernel")
plt.axis("off")
plt.show()
```

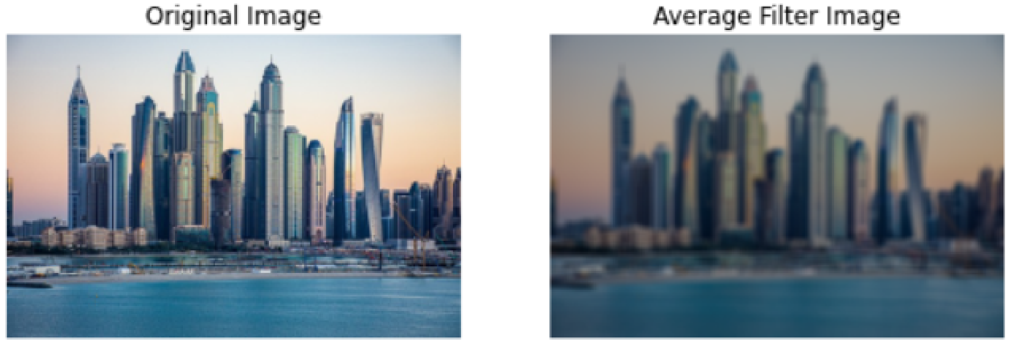
ii) Using Laplacian Operator

```
laplacian=cv2.Laplacian(image2,cv2.CV_64F)
plt.figure(figsize=(9,9))
plt.subplot(1,2,1)
plt.imshow(image2)
plt.title("Original Image")
plt.axis("off")
plt.subplot(1,2,2)
plt.imshow(laplacian)
plt.title("Laplacian Operator")
plt.axis("off")
plt.show()
```

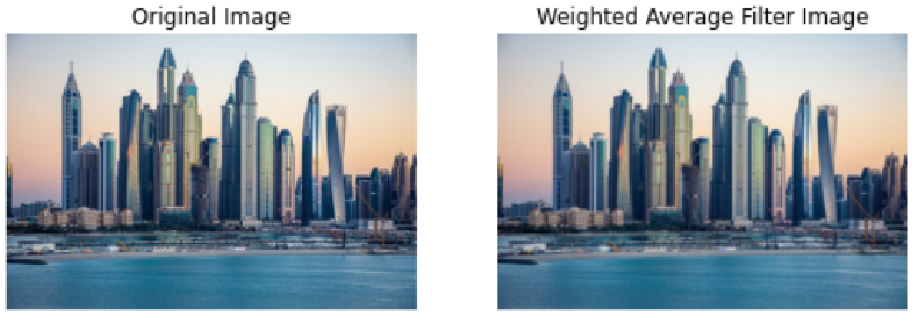
OUTPUT:

1. Smoothing Filters

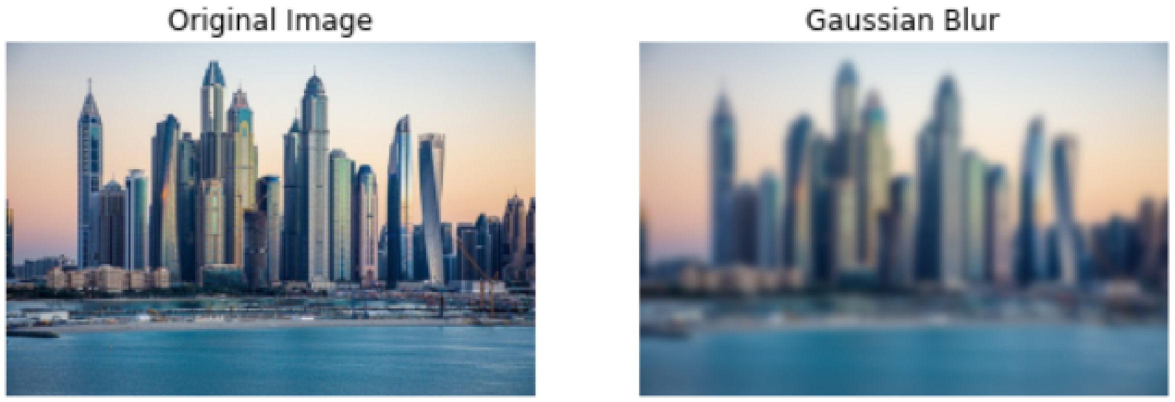
i) Using Averaging Filter



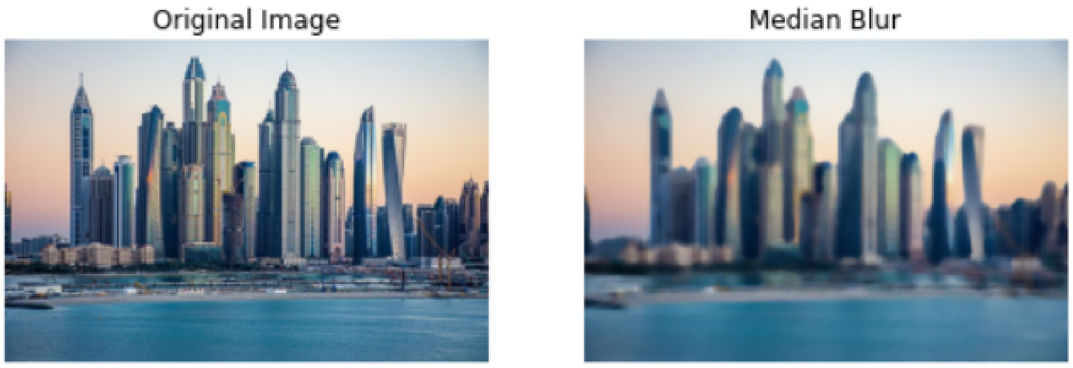
ii) Using Weighted Averaging Filter



iii) Using Gaussian Filter

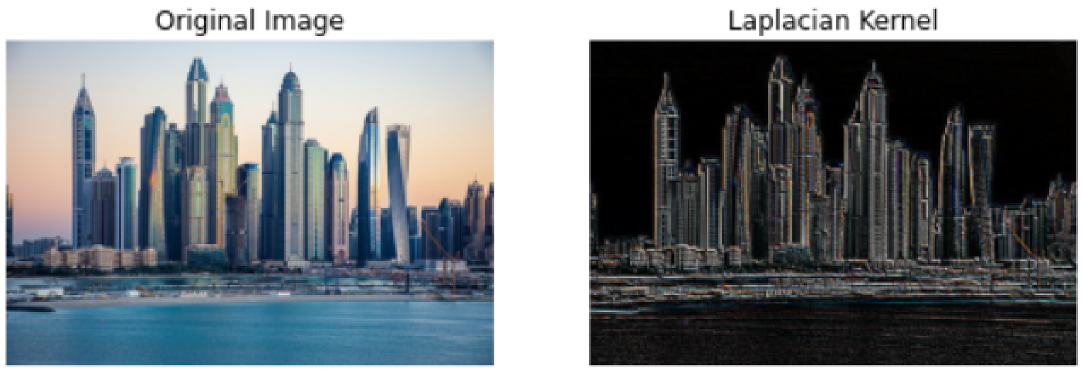


iv) Using Median Filter

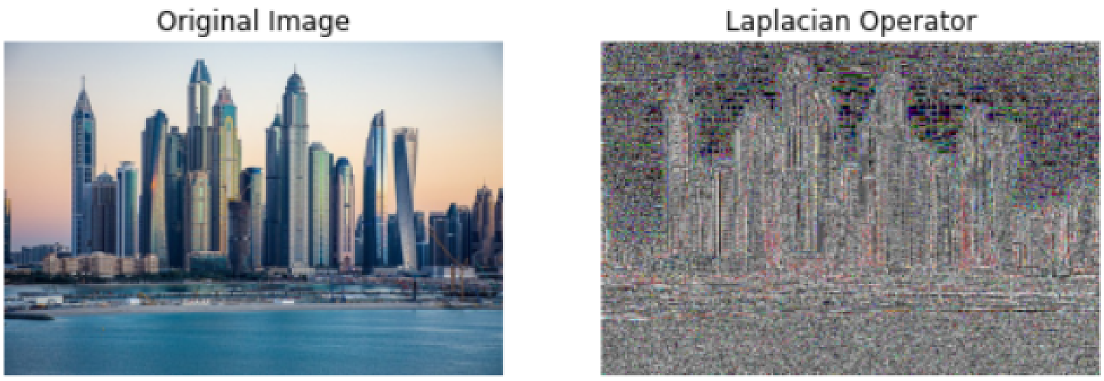


2. Sharpening Filters

i) Using Laplacian Kernal



ii) Using Laplacian Operator



Result:

Thus the filters are designed for smoothing and sharpening the images in the spatial domain.

main Implementation-of-filter / README.md Top
Preview Code Blame Raw