Module 4 Critical Thinking: Mean, Median, and Gaussian Filters

Jordan Washburn

CSC515 - Foundations of Computer Vision

Dr. Jonathan Vanover

April 11th, 2023

Image filtering is necessary for several reasons. First, for noise reduction purposes, images captured by digital cameras or scanners often contain noise due to sensor limitations, transmission errors, or other factors. Filtering can help reduce or eliminate noise, which typically results in better image quality. Next, feature extraction may be needed and filtering can help specify the desired features we want for analysis. Lastly, image filtering facilitates image enhancement which allows us to improve the visual quality of an image by enhancing features such as edges, textures, or corners (Image Processing, n.d.).

OpenCV (Open Source Computer Vision Library) helps us achieve this in Python and other languages. It is supported in Java, Python, and C++. Typically, Gaussian filters are the most common and most preferred filter in most cases (Naveenkumar & Vadivel, 2015). Here, we have an image with impulse noise. Various filters such as median, mean, and a Gaussian filter are applied to the image with kernel sizes of 3x3, 5x5, and 7x7, to achieve different results. For visualization purposes, matplotlib is used and the output images are displayed on a 3x4 grid for easy comparison. The figures below show the code and the output images.

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

img = cv2.imread('/content/drive/MyDrive/Mod4/Mod4CT1.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

kernel_3 = np.ones((3, 3), np.float32) / 9
median_blur_k3 = cv2.medianBlur(img, 3)
mean_k3 = cv2.blur(img, (3, 3))
gaussian_blur_k3_sigma0 = cv2.GaussianBlur(img, (3, 3), 0)
gaussian_blur_k3_sigma3 = cv2.GaussianBlur(img, (3, 3), 3)

kernel_5 = np.ones((5, 5), np.float32) / 25
median_blur_k5 = cv2.medianBlur(img, 5)
mean_k5 = cv2.blur(img, (5, 5))
gaussian_blur_k5_sigma0 = cv2.GaussianBlur(img, (5, 5), 0)
gaussian_blur_k5_sigma3 = cv2.GaussianBlur(img, (5, 5), 3)

kernel_7 = np.ones((7, 7), np.float32) / 49
median_blur_k7 = cv2.medianBlur(img, 7)
mean_k7 = cv2.blur(img, (7, 7))
gaussian_blur_k7_sigma0 = cv2.GaussianBlur(img, (7, 7), 0)
gaussian_blur_k7_sigma3 = cv2.GaussianBlur(img, (7, 7), 3)

titles = ['Median', 'Mean', 'Gauss s0', 'Gauss s3']
kernel_titles = ['K3', 'K5', 'K7']
images = [
    median_blur_k3, mean_k3, gaussian_blur_k3_sigma0, gaussian_blur_k3_sigma3,
    median_blur_k5, mean_k5, gaussian_blur_k5_sigma0, gaussian_blur_k5_sigma3,
    median_blur_k7, mean_k7, gaussian_blur_k7_sigma0, gaussian_blur_k7_sigma3
]

for i in range(12):
    plt.subplot(3, 4, i + 1)
    plt.imshow(images[i])
    plt.title(titles[i % 4] + " " + kernel_titles[i // 4])
    plt.xticks([])
    plt.yticks([])

plt.show()
```

Figure 1: Mean, Median, and Gaussian Filters in Code

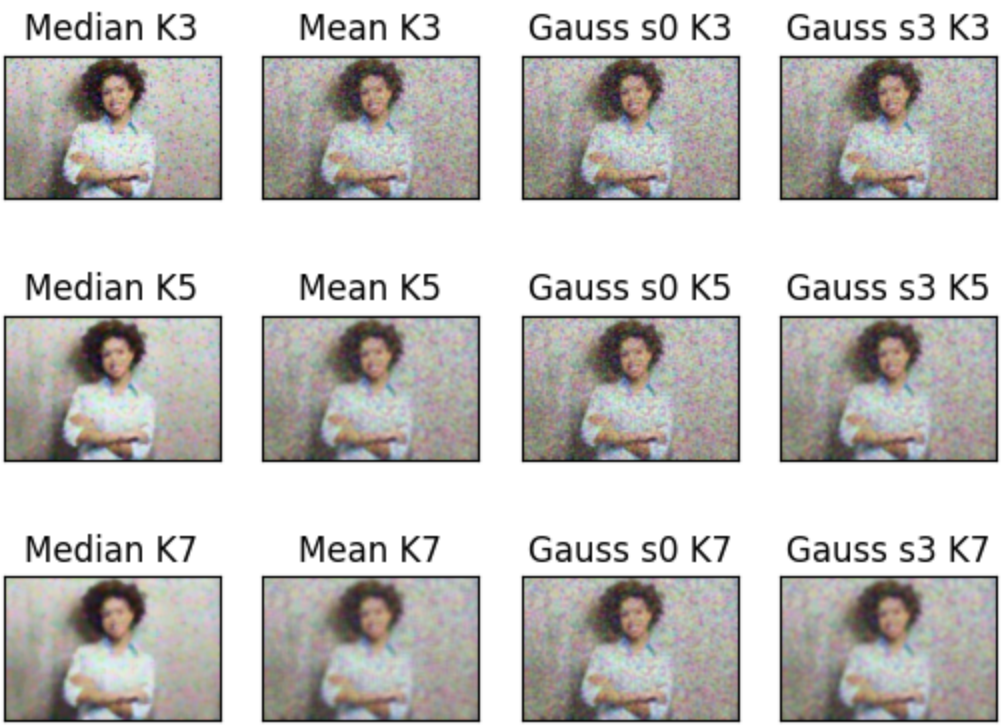

Figure 2: Output Images

We can see that the best performing filter seems to be the median filter with the 5x5 kernel. This filter helps preserve the image features better than the mean filter while still removing a considerable amount of noise. Comparatively, the Guassian filter seems to do fairly well, but does not remove as much of the impulse noise as effectively as the median filter.

This is in line with the expectations. Typically, median or Gaussian filters are the most utilized, however, it depends on the use case. There is not necessarily one “correct” filter for image processing. Each filter serves different functions. For example, Gaussian filters are useful for general-purpose smoothing and blurring, as well as edge preservation. Median filters, on the other hand, are useful for salt-and-pepper noise. Mean filters can be useful in reducing random noise and are often used for general purpose smoothing. However, mean filters may result in too much blur on edges and other important features in the image.

References

Image processing. (2010). Department of Computer Science. https://www.cs.utexas.edu/users/fussell/courses/cs384g-fall2011/lectures/lecture04-Image_Processing.pdf

Naveenkumar, M., & Vadivel, A. (2015, March). OpenCV for computer vision applications. In *Proceedings of national conference on big data and cloud computing (NCBDC'15)* (pp. 52-56).