SciComp with Py

CVIP

Basic Image Processing with OpenCV

Part 06

Vladimir Kulyukin
Department of Computer Science
Utah State University

Review

Blurring

- Blurring is a filtering operation
- A sharp image is an image where one can clearly see all objects
- Sharpness is a consequence of clear edges

Why Blur?

- We may want to blur to make the image smoother (remove some small edges here and there) in the image to make subsequent processing more effective
- We may want to blur to create an artistic effect (e.g., motion blur)

Problem

Write a program that takes a command line argument that specifies a path to an image, applies various blurring filters to the image and displays the results.

Sample run:

```
$ python blurring.py road01.png
```

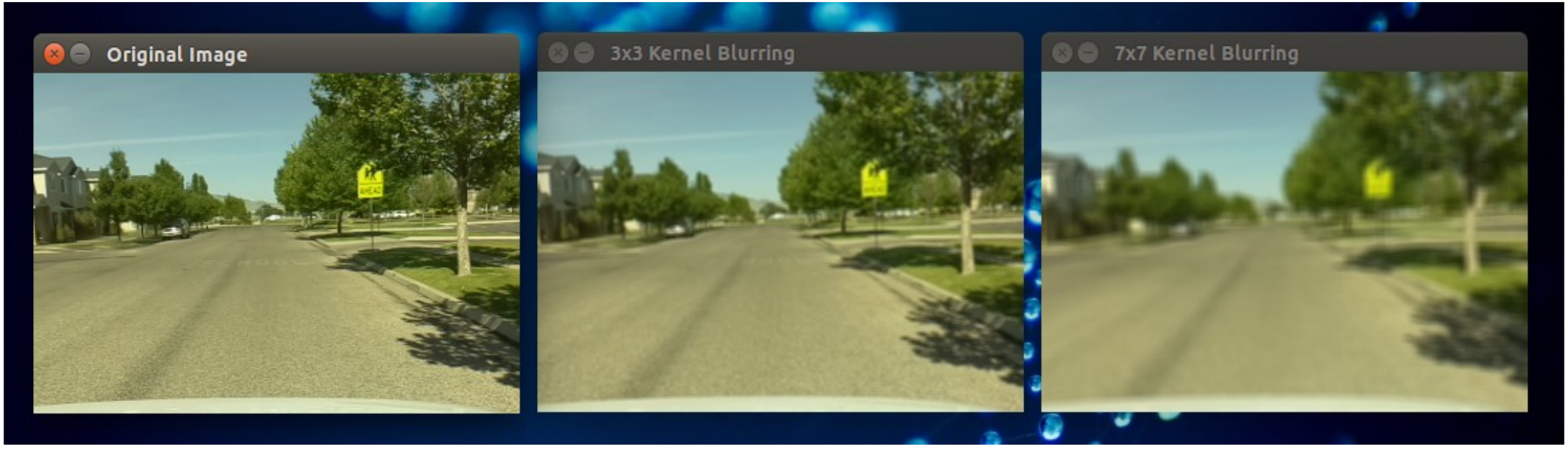
Mean Blurring

```
image = cv2.imread(sys.argv[1])
cv2.imshow('Original Image', image)

kernel_3x3 = np.ones((3, 3), np.float32) / 9

blurred = cv2.filter2D(image, -1, kernel_3x3)
cv2.imshow('3x3 Kernel Blurring', blurred)

kernel_7x7 = np.ones((7, 7), np.float32) / 49

blurred2 = cv2.filter2D(image, -1, kernel_7x7)
cv2.imshow('7x7 Kernel Blurring', blurred2))
```

What is -1? This means the depth (number of bits for each color in a single pixel) of the blurred image (blurred) will be the same as the depth of the original image (image)

source in blurring.py

Sample Run



Gaussian, Mean, and Median Blurring

Gaussian Blurring

Pixel's x, y coordinates

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Sigma is the standard deviation either in a kernel (sometimes in the entire image)

Gaussian, Median, Mean and Bilateral Blur

```
image = cv2.imread(sys.argv[1])
cv2.imshow('Original Image', image)


blur = cv2.blur(image, (3, 3))
cv2.imshow('Mean (3x3)', blur)
gauss = cv2.GaussianBlur(image, (7, 7), 0)
cv2.imshow('Gaussian (7x7)', gauss)
median = cv2.medianBlur(image, 5)
cv2.imshow('Median (5x5)', median)
## bilateral is great for keeping edges sharp.
bilateral = cv2.bilateralFilter(image, 9, 75, 75)
cv2.imshow('Bilateral 9', bilateral)
```

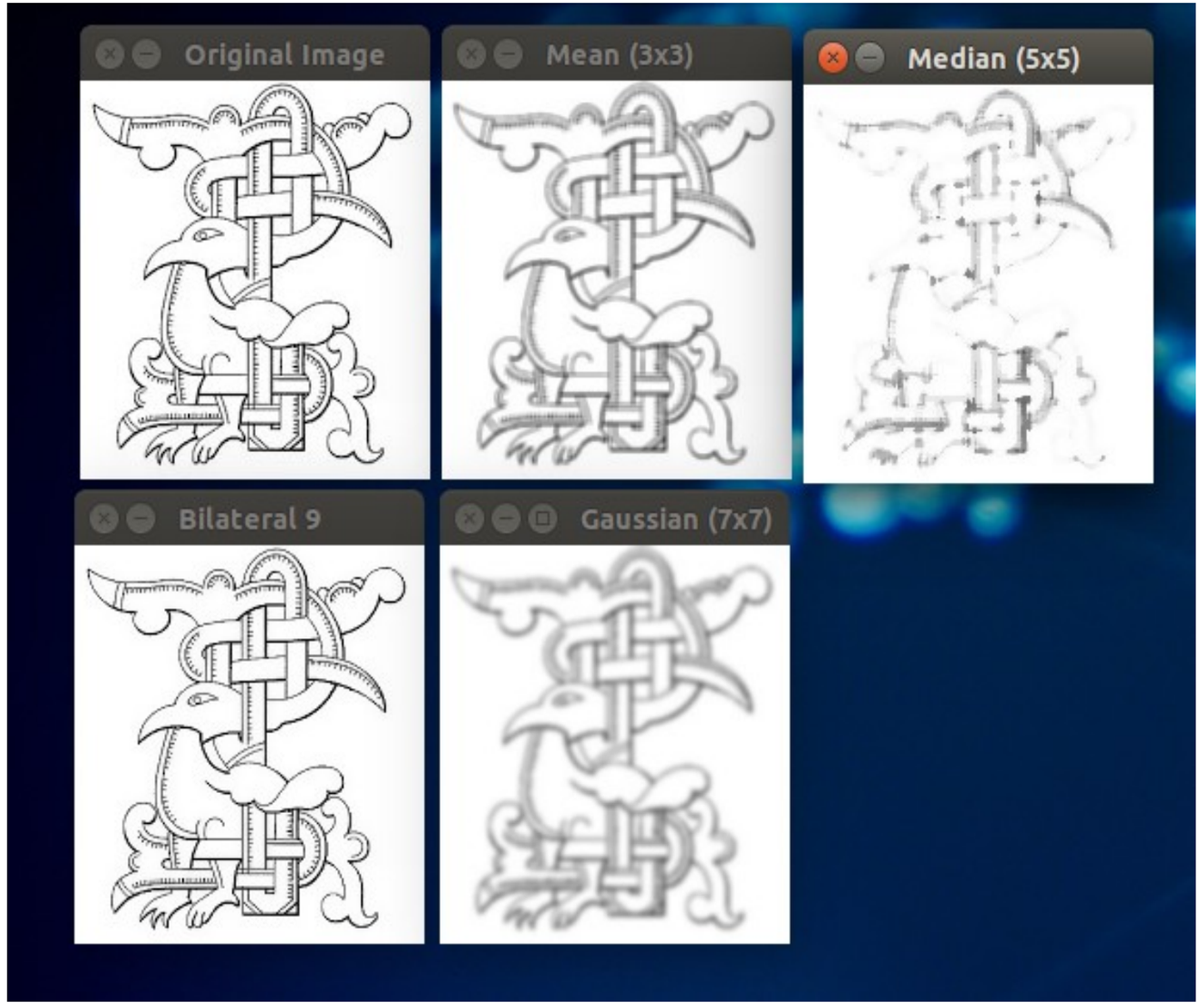
Replaces the pixel in the center of a 5x5 kernel with the median value of the kernel's pixels

Similar to Gaussian but keeps edges sharper

Test Run


Original Image
Mean (3x3)
Median (5x5)
Gaussian (7x7)
Bilateral 9

Test Run


Original Image
Mean (3x3)
Median (5x5)
Bilateral 9
Gaussian (7x7)

Erosion and Dilation

Erosion & Dilation

- Two most common morphological filters are **erosion** and **dilation**
- Erosion replaces the current pixel with the minimum pixel value found in the kernel (e.g., 3 x 3 mask)
- Dilation replaces the current pixel with the maximum pixel value found in the kernel

Erosion & Dilation

- Let us suppose that we apply erosion and dilation to a binary image (**0 – black**, **255 – white**)
- We expect erosion to increase the amount of blackness in the image (since the minimum pixel value is chosen in each shape element)
- We expect dilation to increase the amount of whiteness in the image (since the maximum pixel value is chosen in each shape element)

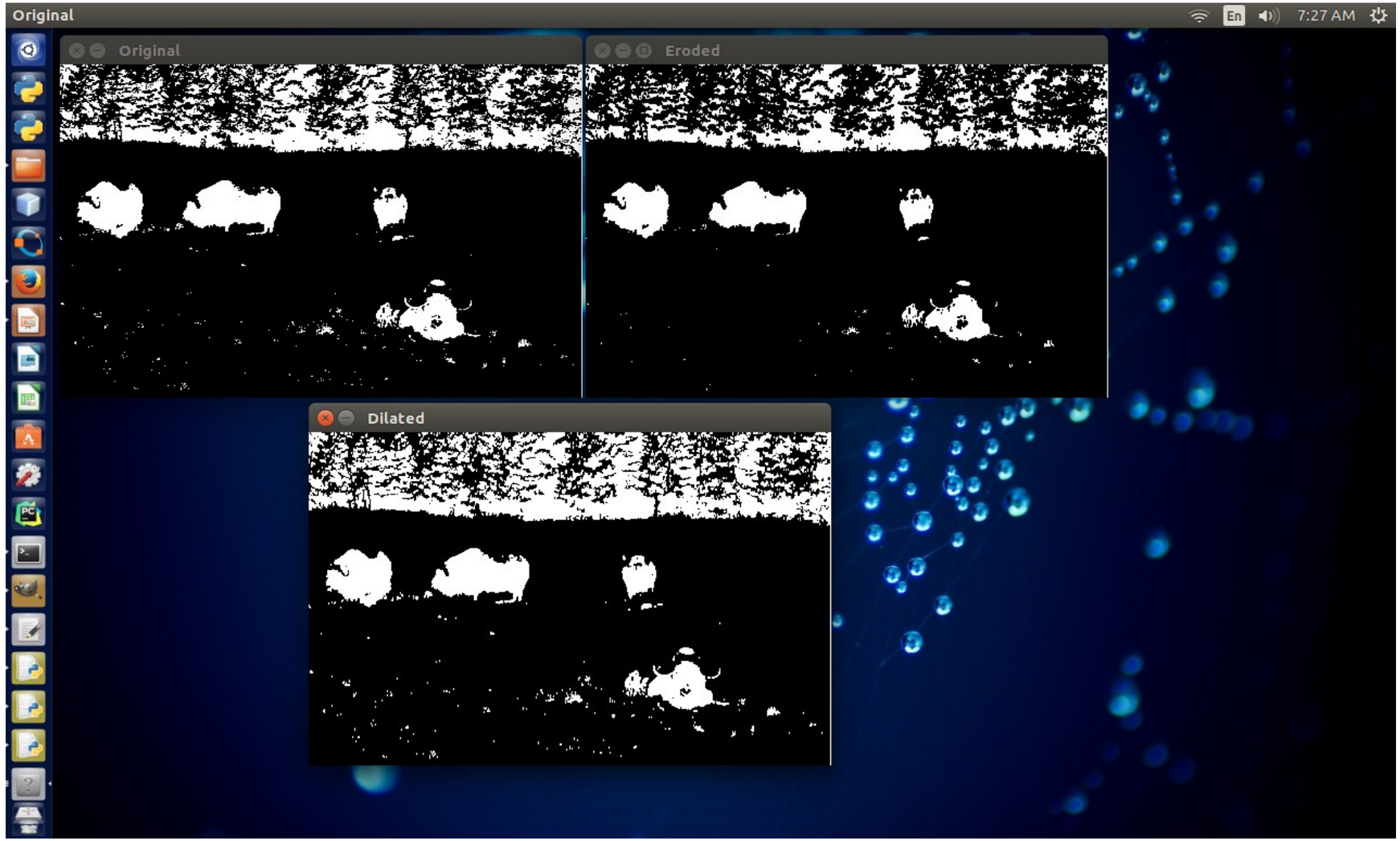
Erosion & Dilation

Image Source: R. Laganiere. “**OpenCV 2 Cookbook**”, Ch. 05

Erosion & Dilation

```
import cv2
import sys

img = cv2.imread(sys.argv[1])
cv2.imshow('Original', img)

er_img = cv2.erode(img, (5, 5))
cv2.imshow('Eroded', er_img)

dl_img = cv2.dilate(img, (5, 5))
cv2.imshow('Dilated', dl_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Test Run



References

- https://en.wikipedia.org/wiki/Gaussian_blur
- https://en.wikipedia.org/wiki/Erosion_(morphology)
- https://en.wikipedia.org/wiki/Dilation_(morphology)
- www.opencv.org