SciComp with Py

CVIP

Pixel Masks, Blur, Erosion, Dilation

Vladimir Kulyukin
Department of Computer Science
Utah State University

Outline

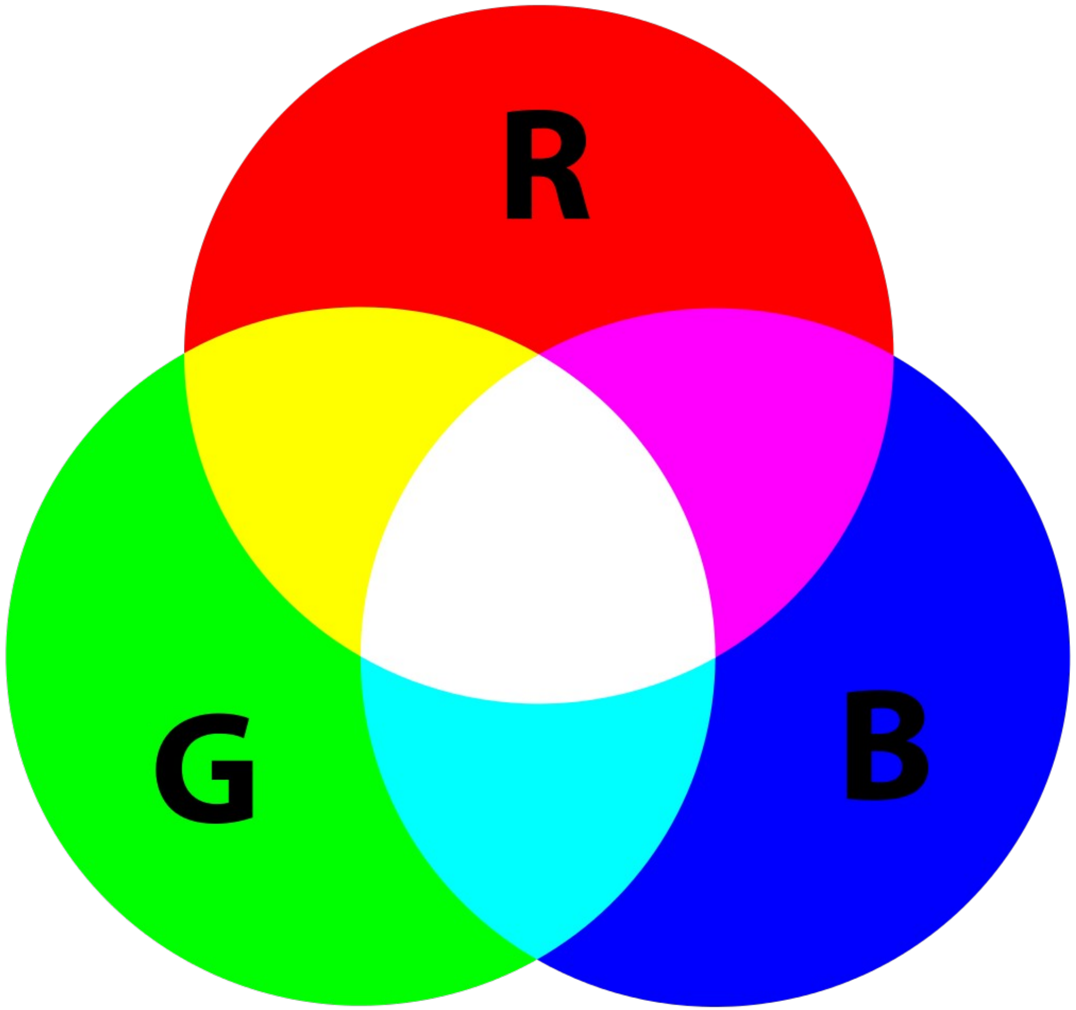
- Review
- Pixel Masks
- Blur, Erosion, Dilation

Review

3 Big M's

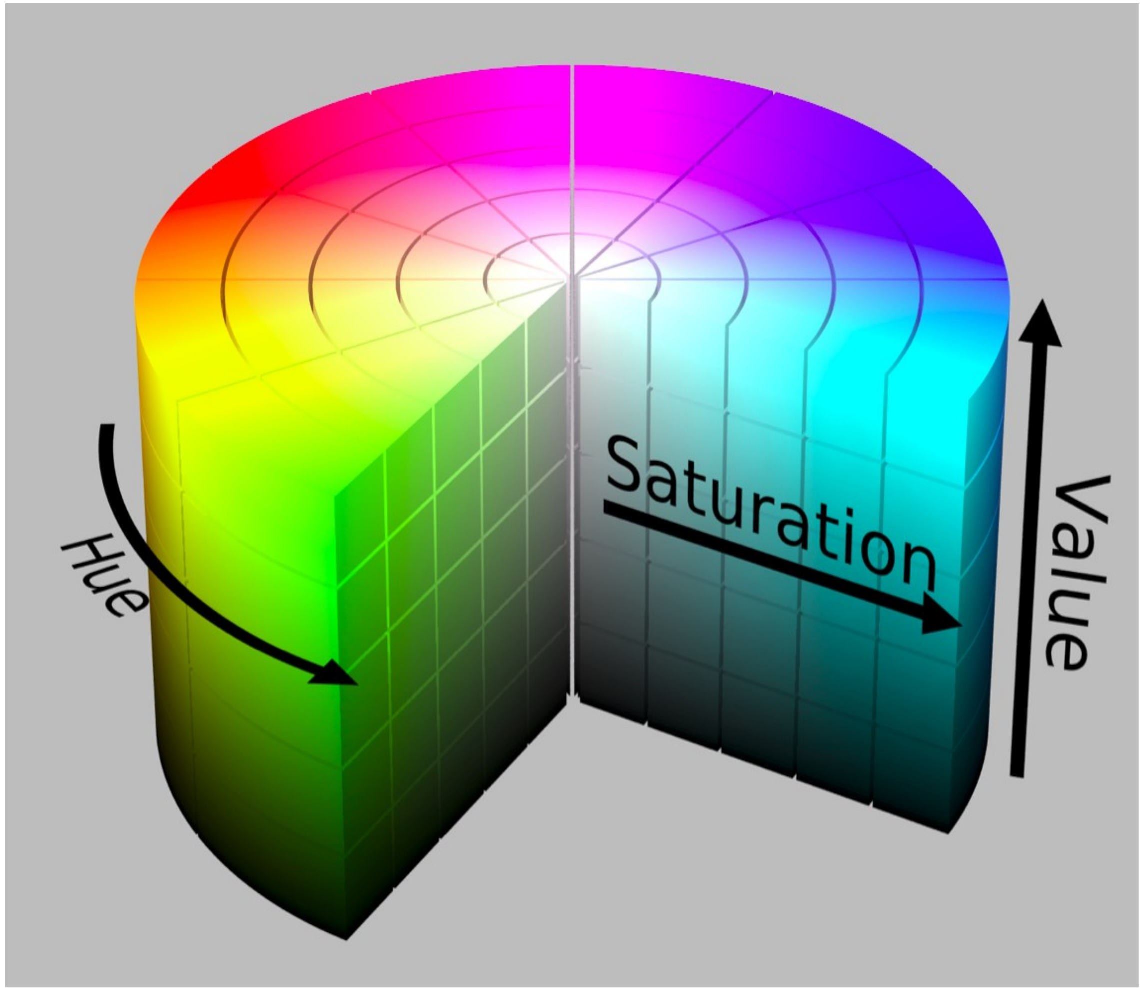
- Mean – is the average of a set of values
- Median – a numerical value **v** right in the middle of the sorted sequence of values so that exactly half of the values in the set are less than v and half are greater than **v**
- Mode – the most frequent value in a set of values

RGB Color Space



In OpenCV, the standard pixel representation is B, G, R, e.g., [10, 234, 50]. The values of B, G, R are in [0, 255]. [0, 0, 0] is black; [255, 255, 255] is white.

HSV Color Space



- Hue is color value [0, 179]
- Saturation is color vibrancy [0, 255]; at lower saturation in the center everything is white
- Value is brightness/intensity of color [0, 255]; it goes from dark (below) to bright (above)

Accessing Columns in 3D Arrays

```
>>> m = np.random.rand(2, 2, 3)
>>> m
array([[[ 0.5828791 ,  0.6274894 ,  0.28059109],
        [ 0.1992719 ,  0.90985773,  0.65487014]],
       [[ 0.71503477,  0.84349841,  0.97827513],
        [ 0.69862272,  0.19670252,  0.82069481]]])
>>> m[:, :, 0]
array([[ 0.5828791 ,  0.1992719 ],
       [ 0.71503477,  0.69862272]])
>>> m[:, :, 1]
array([[ 0.6274894 ,  0.90985773],
       [ 0.84349841,  0.19670252]])
>>> m[:, :, 2]
array([[ 0.28059109,  0.65487014],
       [ 0.97827513,  0.82069481]])
```

getting column 0 numbers

getting column 1 numbers

getting column 2 numbers

Problem

Write a program that converts an image from RGB to HSV, displays the HSV image, and displays each channels of the HSV image (H, S, and V) in separate images.

Sample Call

```
$ python   bgr_to_hsv.py  -i truck.jpg
```

Problem

Write a program that takes an image and prints out the pixel values in a given row or columns.

Sample Calls

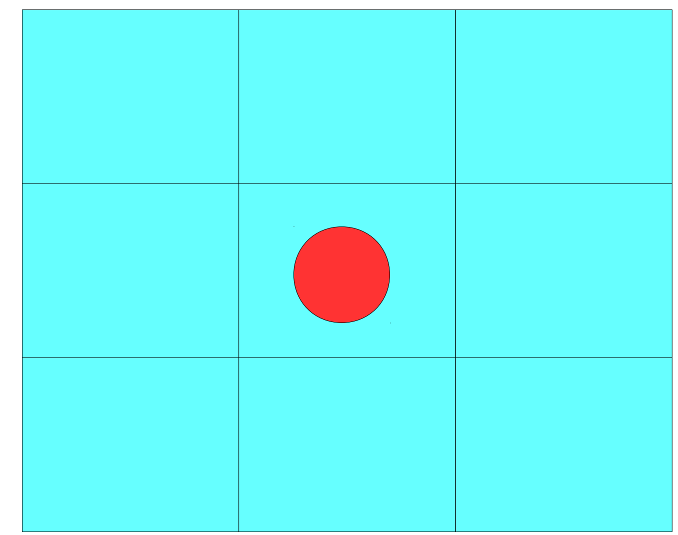
```
$ python   display_row.py -i verline.png -r 10
$ python   display_col.py -i verline.png -c 10
$ python   display_row.py -i verline.png -c 80
```

Pixel Masks

Pixel Masks

	0	1	2	…	N
0	128	255	10	201	203
1	120	35	50	25	137
2	34	89	190	197	108
…	180	178	215	37	45
M	24	25	91	225	225

3 x 3 Pixel Mask



Pixel masks are used to compute various properties of the center pixel

Pixel Masks

	0	1	2	...	N
0	128	255	10	201	203
1	120	35	50	25	137
2	34	89		197	108
...	180	178	215	37	45
M	24	25	91	225	225

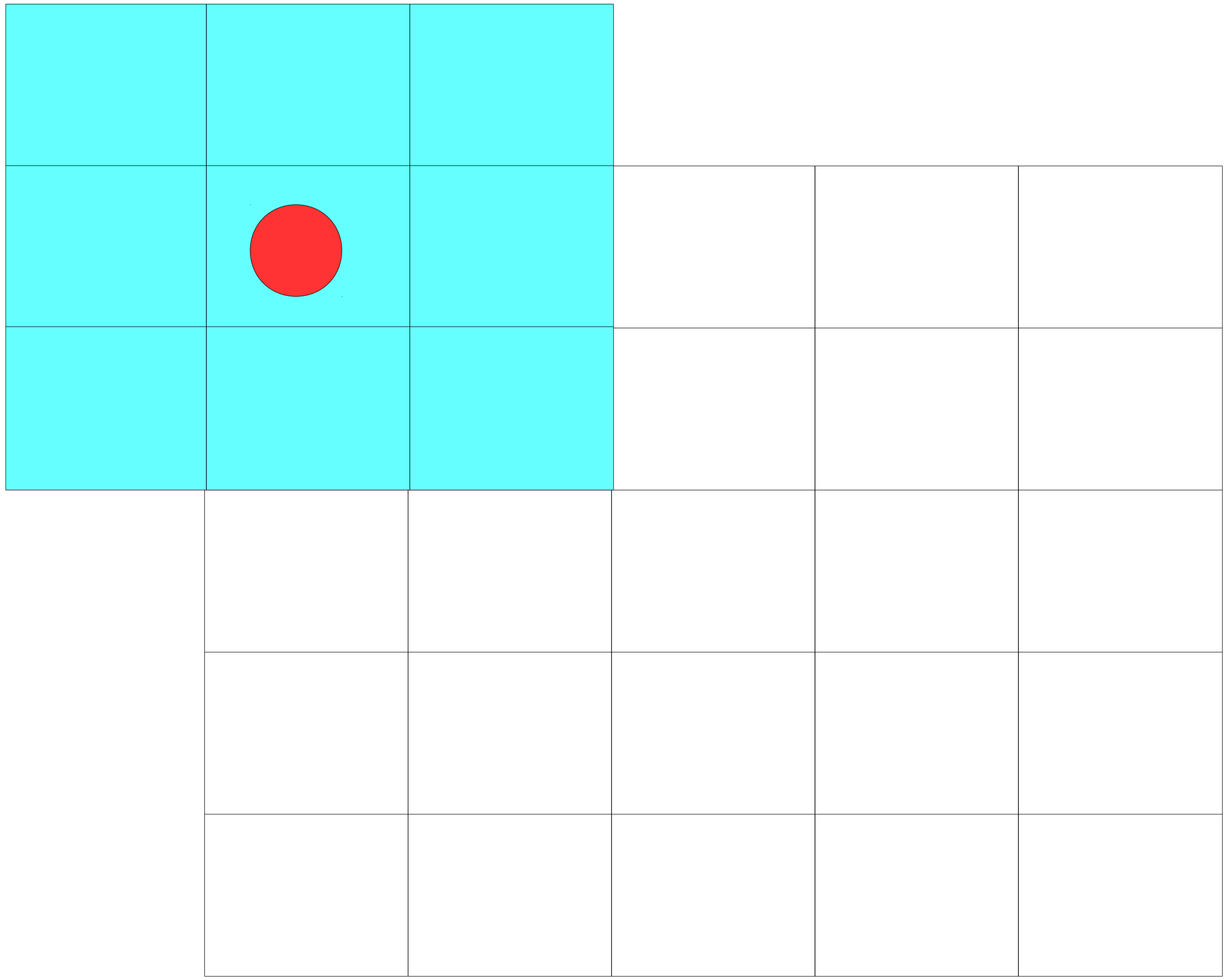
Pixel masks are superimposed on the image to compute various properties of the center pixel.

Computing Cell Properties

	0	1	2	3	4
0	128	255	10	201	203
1	120	35	50	25	137
2	34	89		197	108
3	180	178	215	37	45
4	24	25	91	225	225

Properties of cell I[2, 2] can be computed in terms of cells I[1,1], I[2, 1], I[3, 1], I[1, 2], I[3,2], I[1, 3], I[2, 3], and I[3,3].

Border Pixel Problem



What happens when the mask is centered at a border pixel? Some pixels covered by the mask do not exist.

Two possible approaches:
1) Pad the image; this can be done virtually.
2) Do not center the mask at the border pixels.

Image Convolution: Applying Masks to Images

- Given an image **I** and a mask **M**, **M** is centered at each possible pixel and a value **v** is computed
- This value **v** is saved in an new image or the value of the current pixel on which **M** is centered is destructively modified with **v**
- This process is sometimes called *image convolution*

Blur, Erosion, Dilation

Blurring

- Blurring is another type of filtering operation
- A sharp image is an image where one can clearly see all objects
- Sharpness is a consequence of clear edges
- Why do we need to 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)

Types of Blurring

- Mean filter
- Weighted average filter
- Median filter
- Gaussian filter
- Bilateral filter (like Gaussian but keeps edges sharp)
- All these filters (and many more) are available in OpenCV

Gaussian Blurring

Pixel's x, y coordinates

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

Standard deviation either in a kernel or entire image

Problem

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

Sample run:

$ python blurring.py road01.png

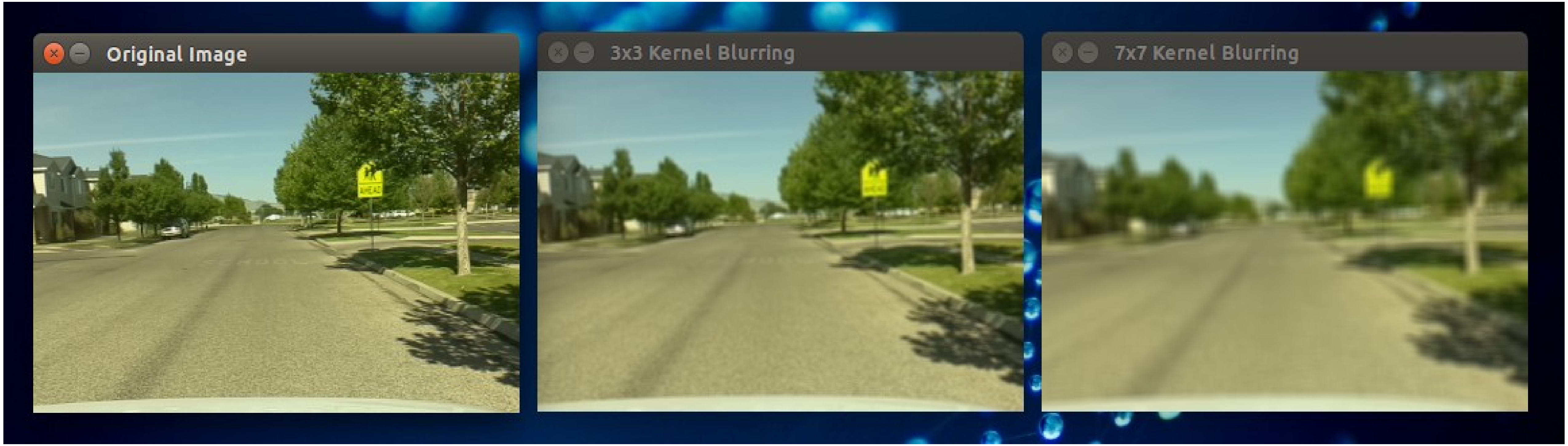
Solution

```
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



Solution

blurring2.py

```
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)
```

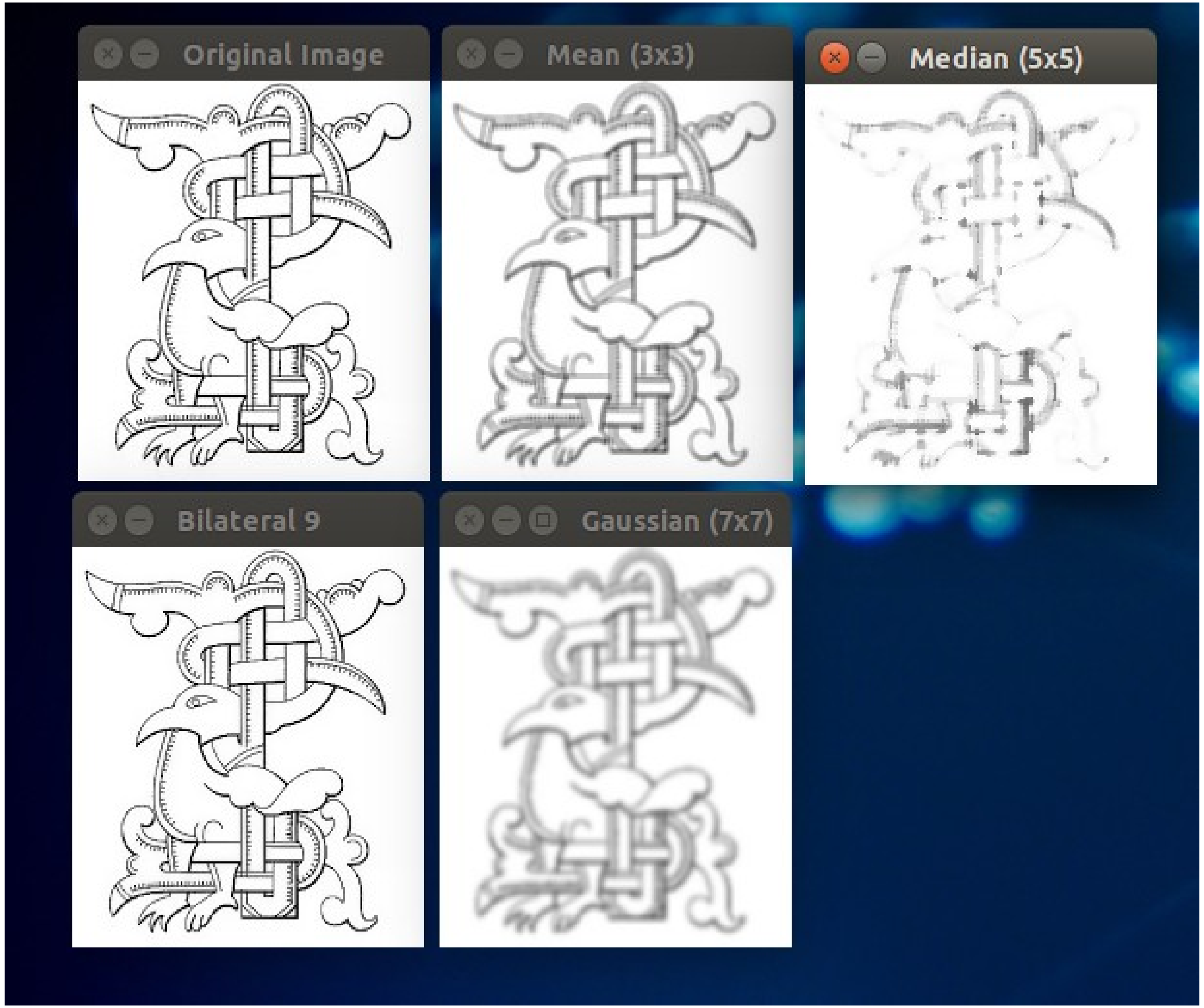
Replaces the pixel in the center of a 5x5 kernel with the median value of the kernel's pixels

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 & Dilation

Erosion & Dilation

- Two most common morphological filters are **erosion** and **dilation**
- Erosion replaces the current pixel with the minimum pixel found in the kernel
- Dilation replaces the current pixel with the maximum pixel 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)

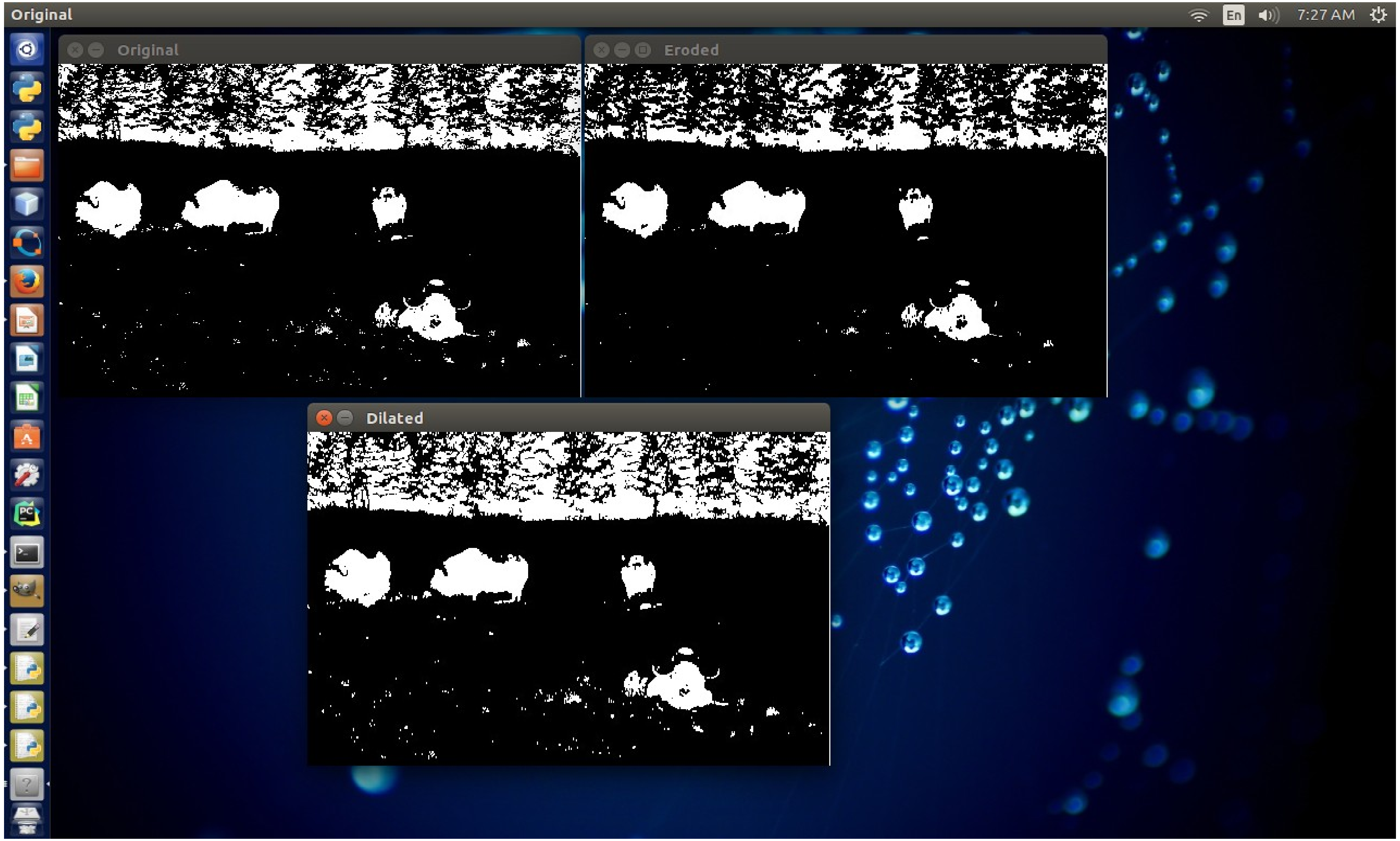
Erosion & Dilation

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