

ENPM 809T

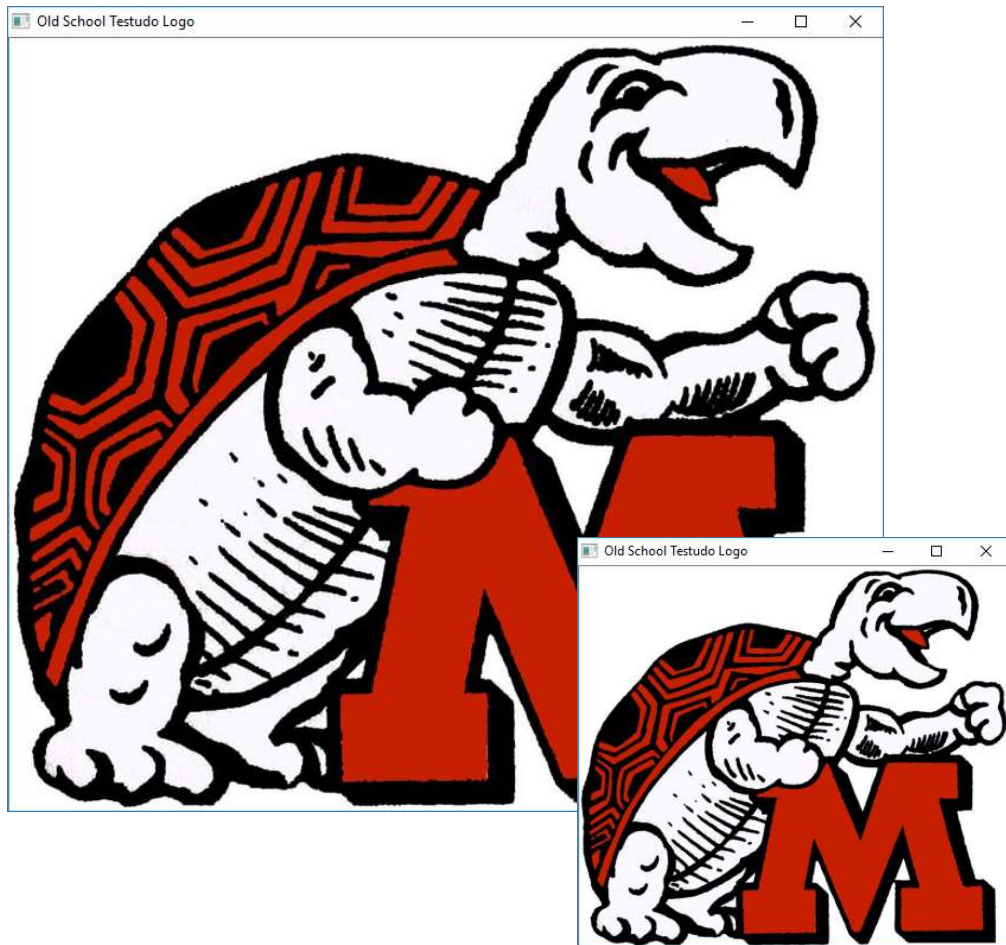
UMCP, Mitchell, Summer 2019

Disclaimer

- Much of the code provided in the following slides was developed in Python **2.7**

Confirm Installation

- Run sanitycheck.py



```
import numpy as np
import matplotlib
import matplotlib.pyplot as plot
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

cv2.imshow("Old School Testudo Logo", image)
cv2.waitKey(0)

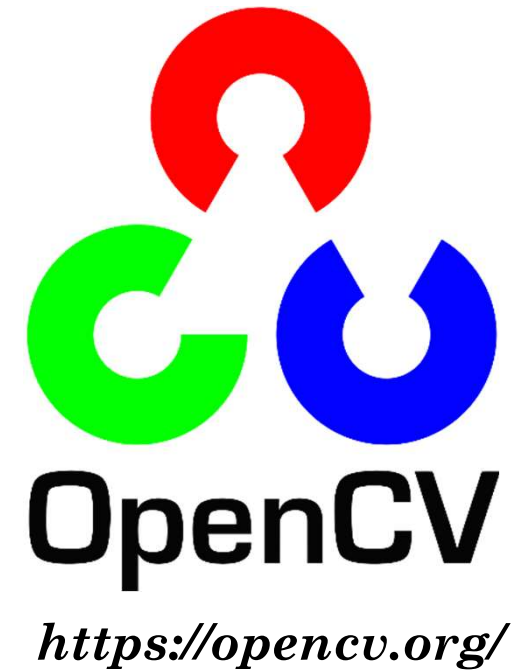
image = imutils.resize(image, width=400)

cv2.imshow("Old School Testudo Logo", image)
cv2.waitKey(0)

cv2.imwrite("testimage.jpg", image)
```

OpenCV

- Open source Computer Vision
- Library of programming functions aimed at real-time computer vision
- Intel, 1999
- Written in C++, binding in Python



Imutils

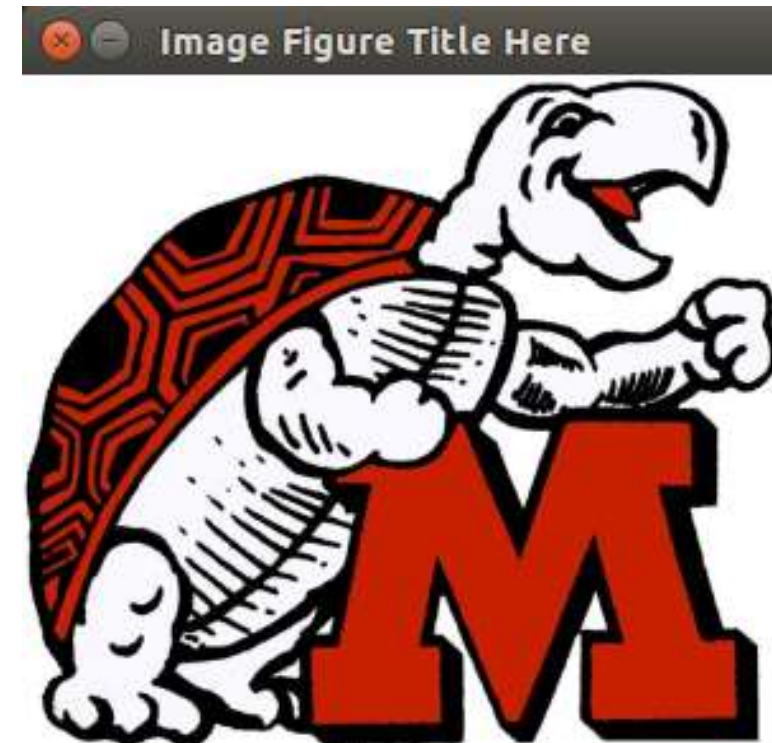
- Series of OpenCV & convenience functions that perform basic tasks
 - Translation
 - Rotation
 - Resizing
- Adrian Rosebrock, 2015



<https://github.com/jrosebr1/imutils>

OpenCV Fundamentals

- Load image from disk
- Display image to screen
- Write (save) image to disk



Codes available via GitHub - <https://github.com/oneshell/enme809T>

OpenCV Fundamentals

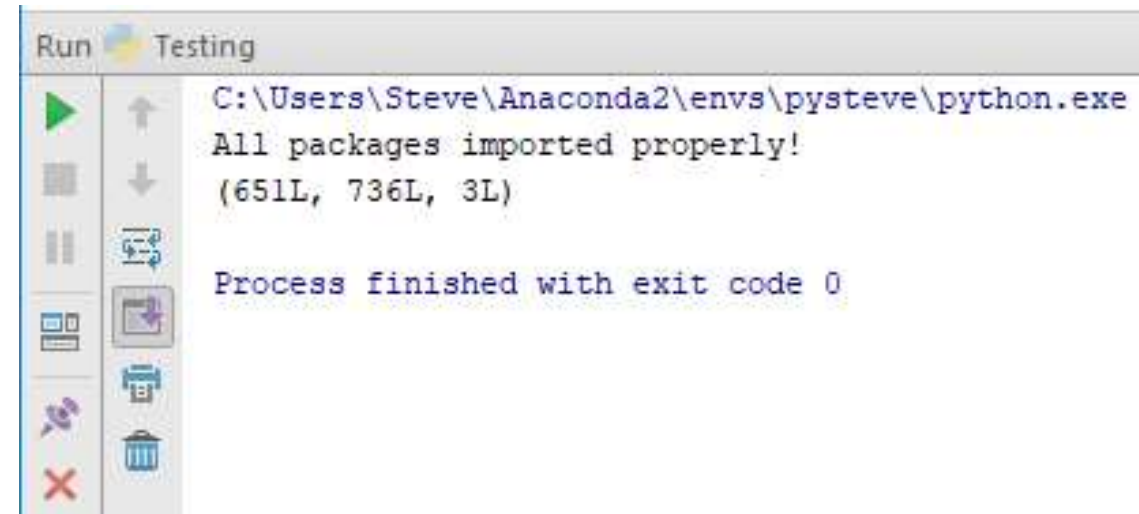
- Images are represented as numpy arrays
- Each array has a shape

```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

print image.shape
```



OpenCV Fundamentals

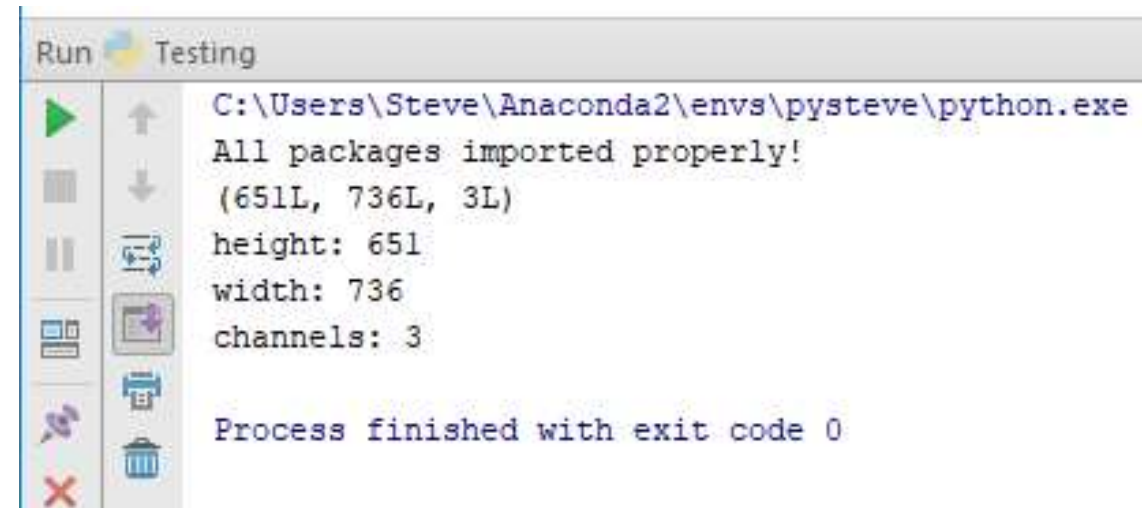
- Access image height, width, and number of channels

```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

print image.shape
print "height: %d" % (image.shape[0])
print "width: %d" % (image.shape[1])
print "channels: %d" % (image.shape[2])
```



OpenCV Fundamentals

- Display image on screen

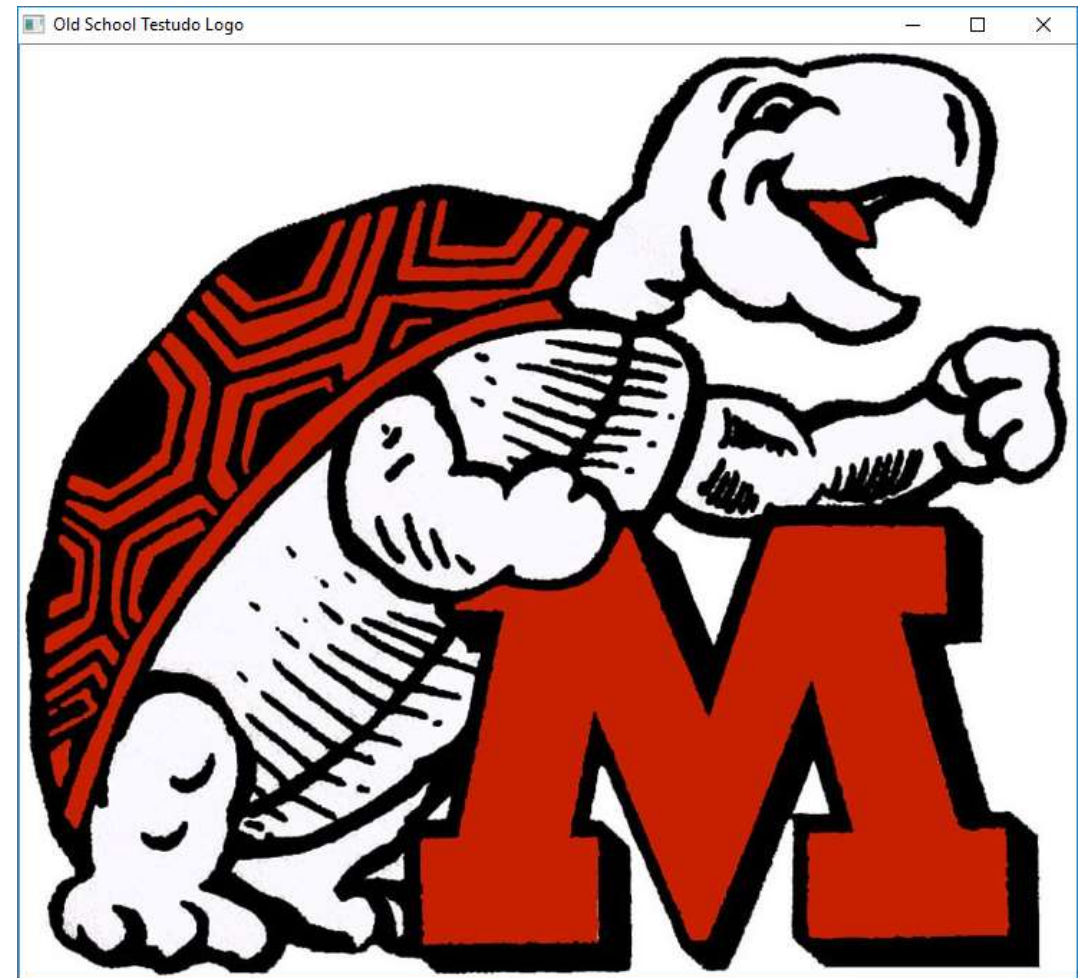
```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

print image.shape
print "height: %d" % (image.shape[0])
print "width: %d" % (image.shape[1])
print "channels: %d" % (image.shape[2])

cv2.imshow("Old School Testudo Logo", image)
cv2.waitKey(0)
```



OpenCV Fundamentals

- Resize image

```
import numpy as np
import cv2
import imutils

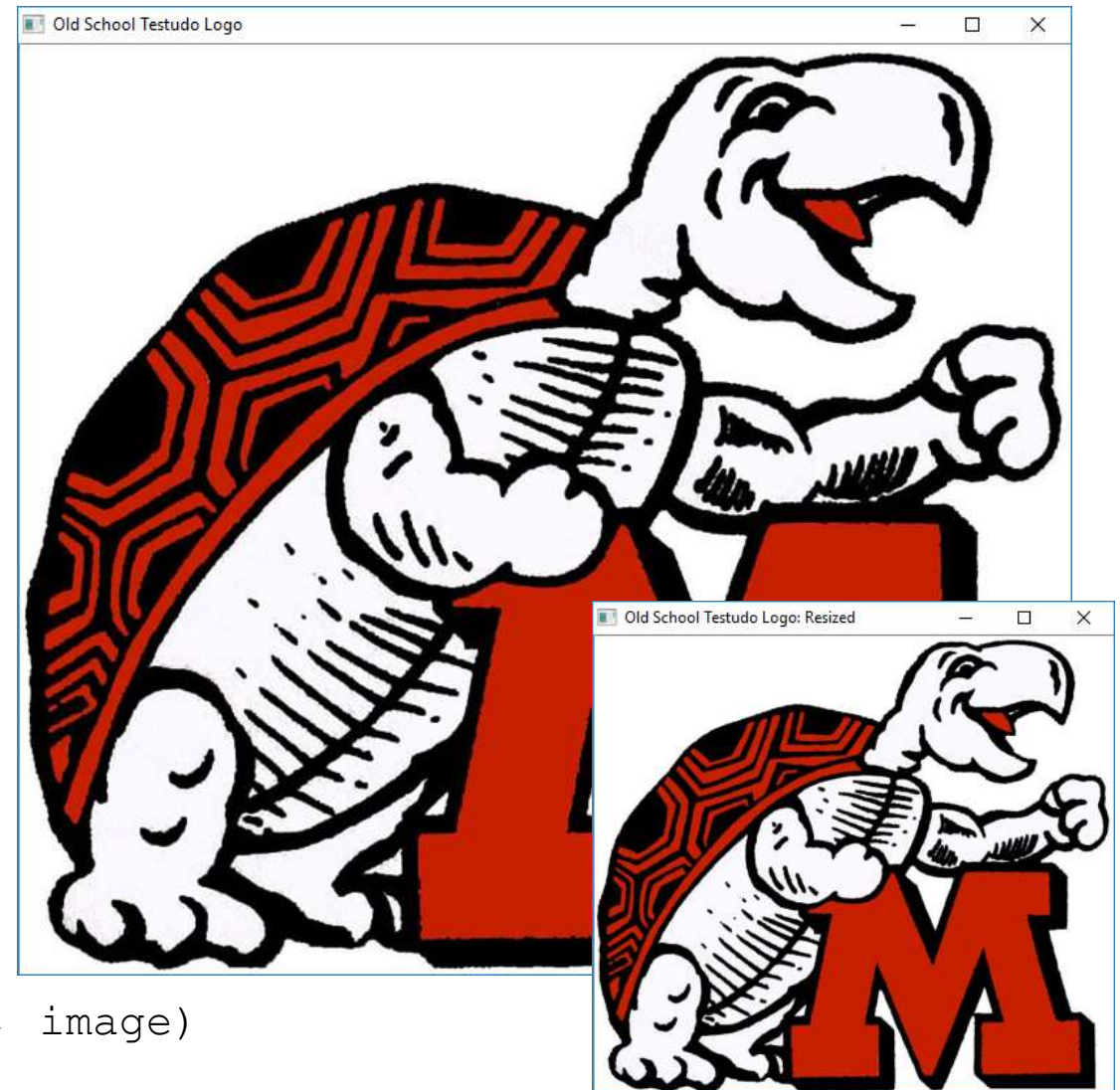
print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

print image.shape
print "height: %d" % (image.shape[0])
print "width: %d" % (image.shape[1])
print "channels: %d" % (image.shape[2])

cv2.imshow("Old School Testudo Logo", image)
cv2.waitKey(0)

image = imutils.resize(image, width=400)
cv2.imshow("Old School Testudo Logo: Resized", image)
cv2.waitKey(0)
```



OpenCV Fundamentals

- Write (save) image to disk

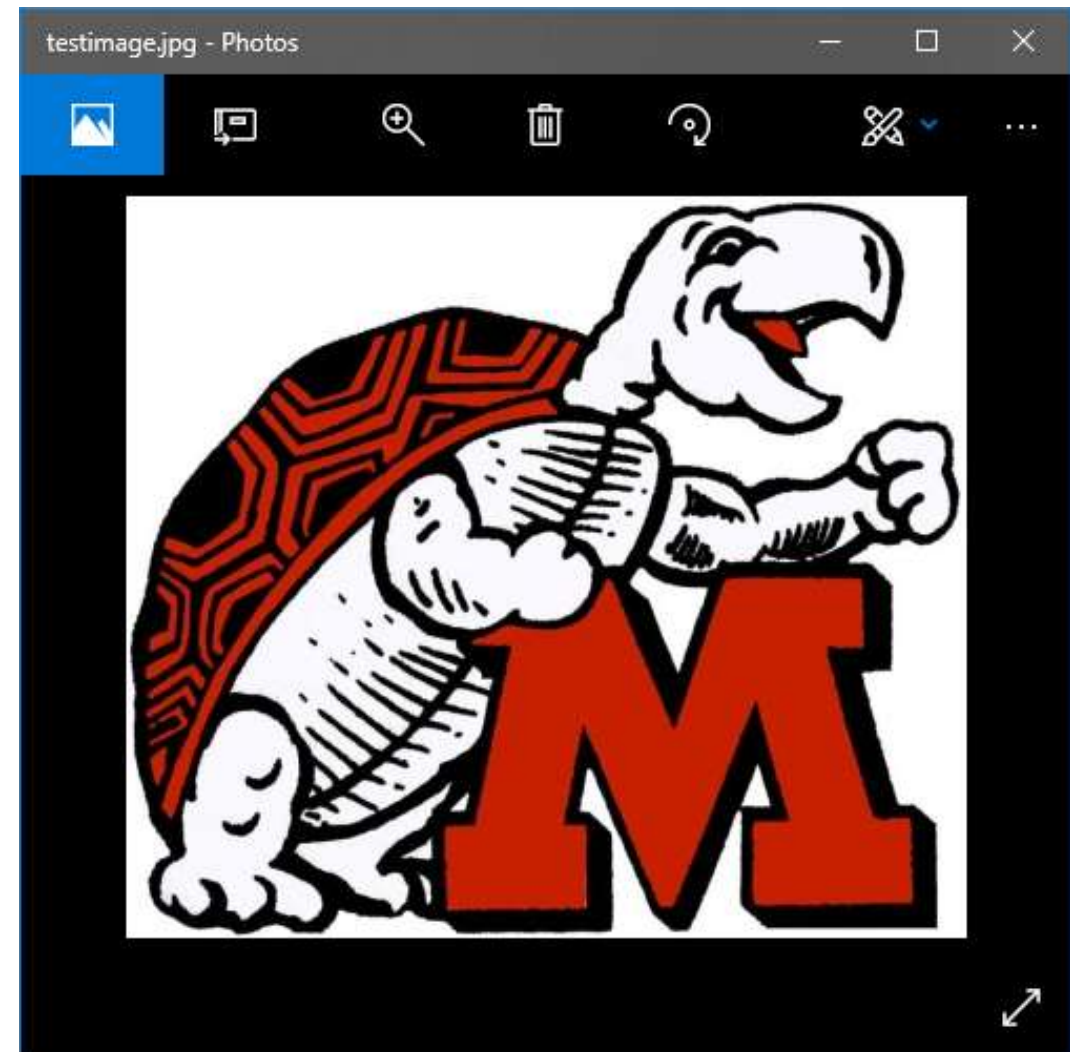
```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

print image.shape
print "height: %d" % (image.shape[0])
print "width: %d" % (image.shape[1])
print "channels: %d" % (image.shape[2])

cv2.imwrite("testimage.jpg", image)
```



OpenCV Fundamentals

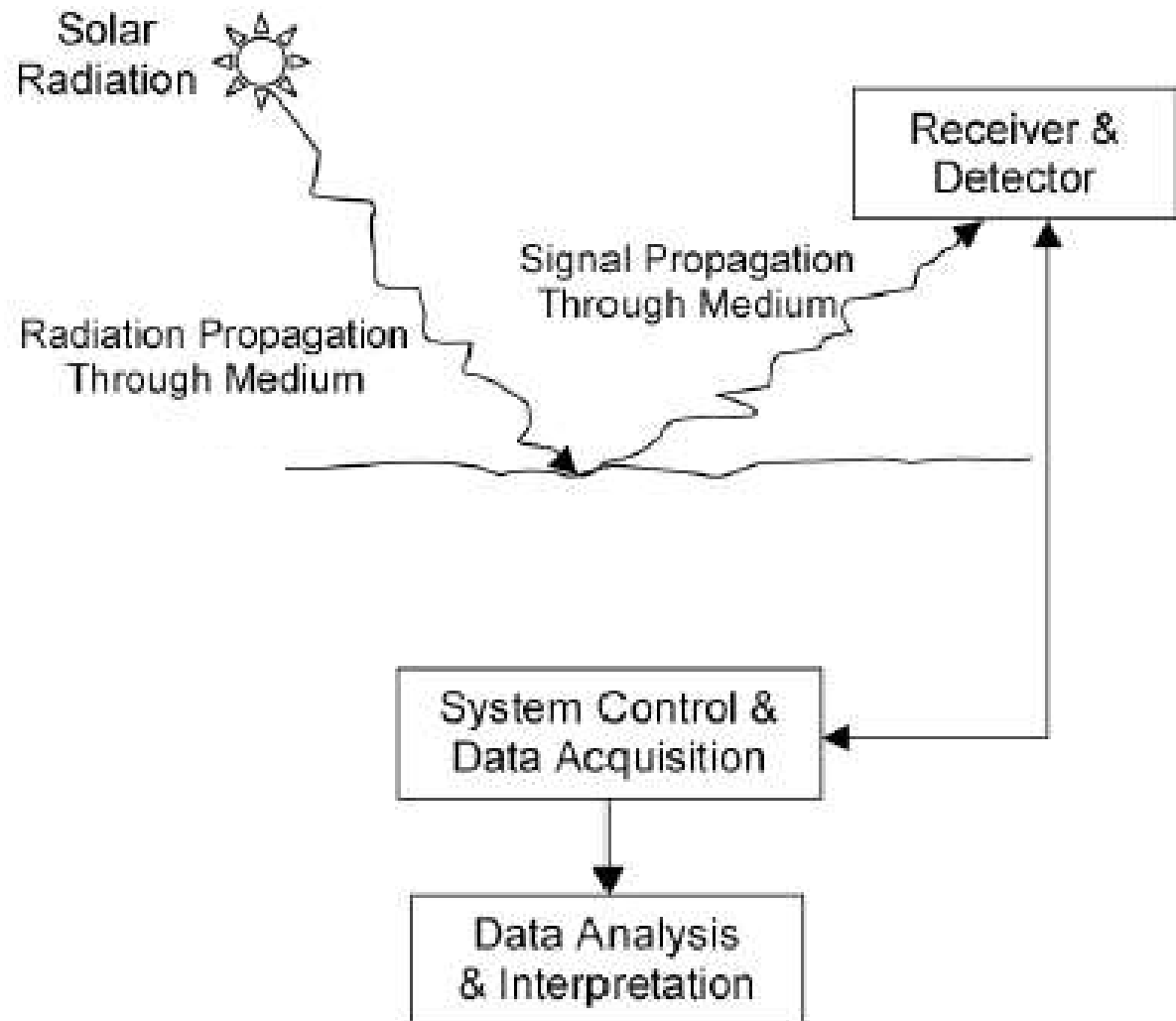
- Pixels
- Defining colors in RGB color space
- Image coordinate system
- Identifying and modifying pixel values

OpenCV Fundamentals

- What is an image?

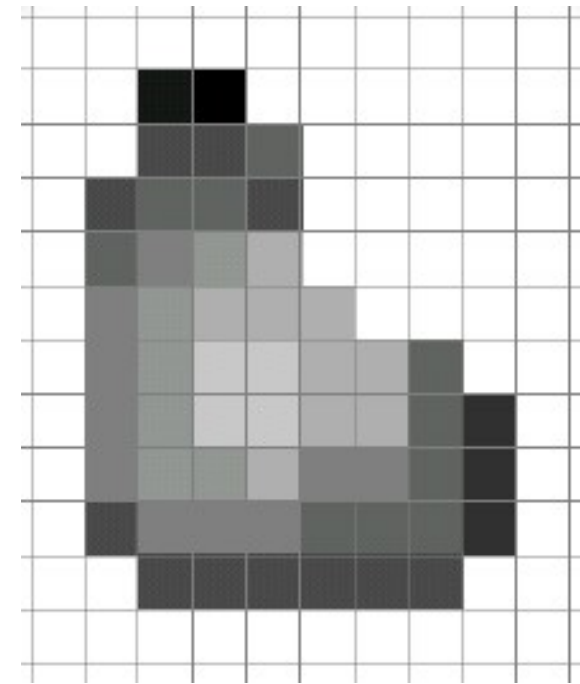
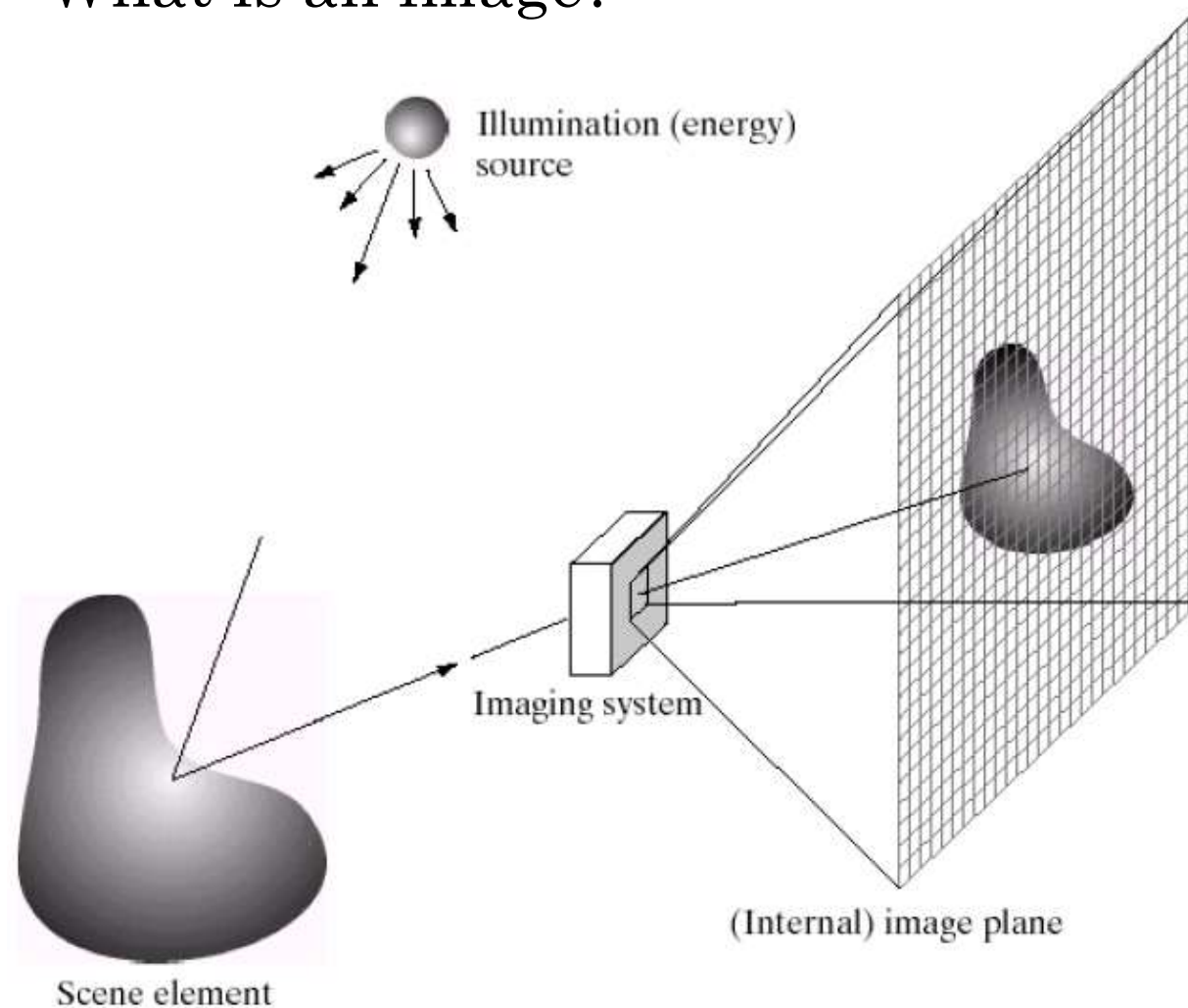
Passive Remote Sensing

- No self-generated radiation is used in the sensing process
- **Naturally occurring radiation** such as sunlight or nightglow emission
- **Photography**
- Spectrometer



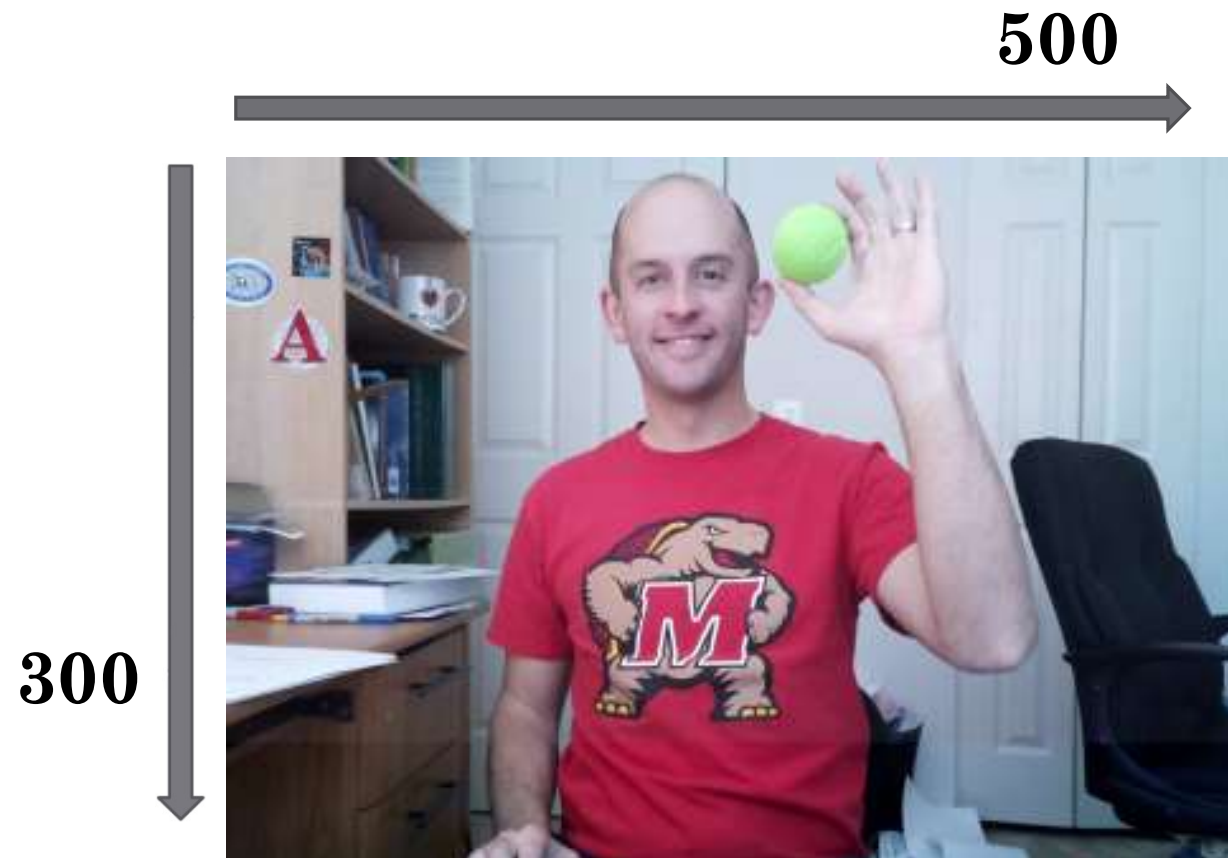
OpenCV Fundamentals

- What is an image?



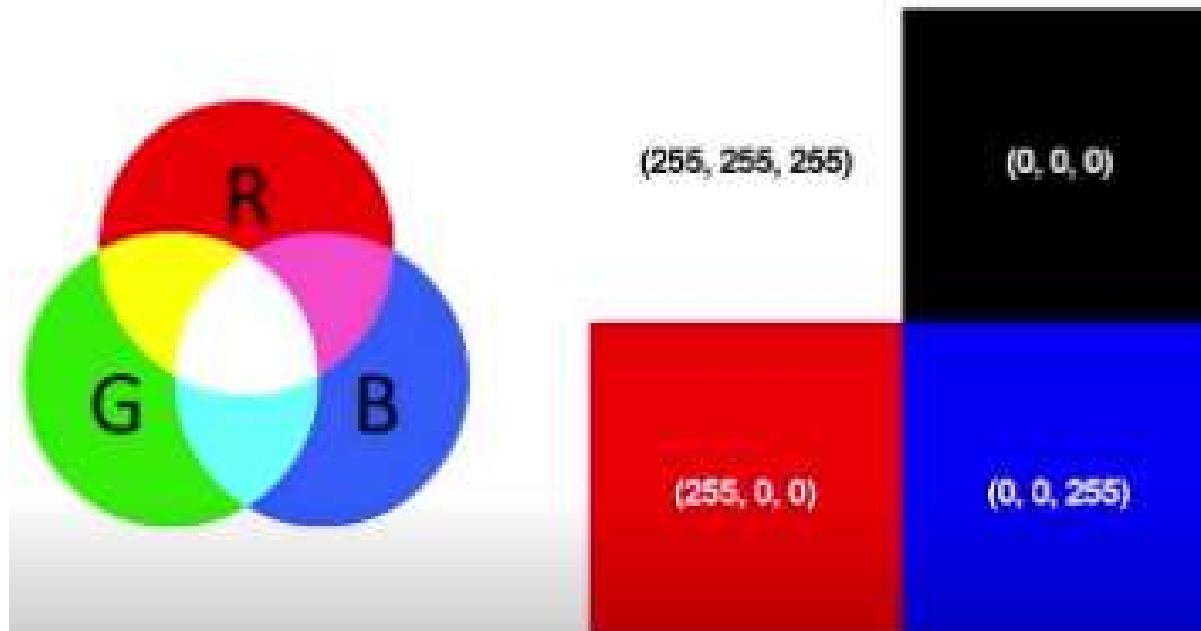
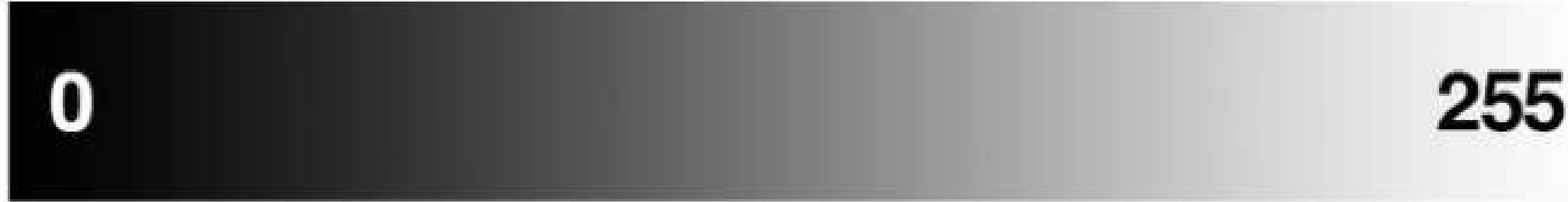
OpenCV Fundamentals

- Pixels: **building blocks** of an image
- No finer granularity than a pixel

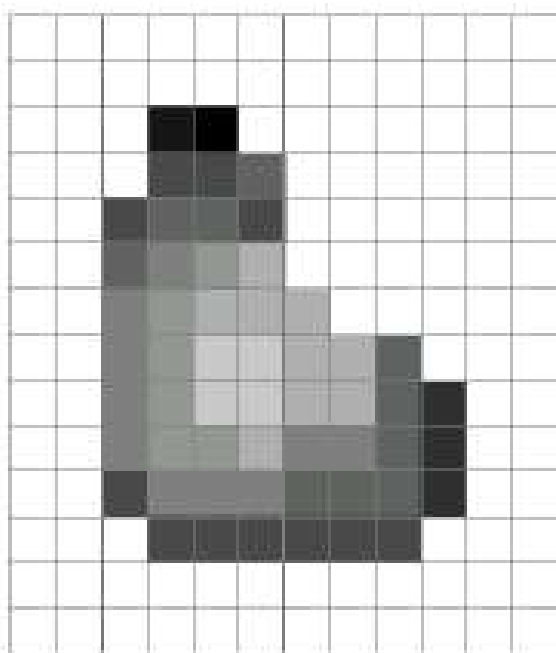


OpenCV Fundamentals

- Pixels typically defined in either grayscale or color



OpenCV Fundamentals



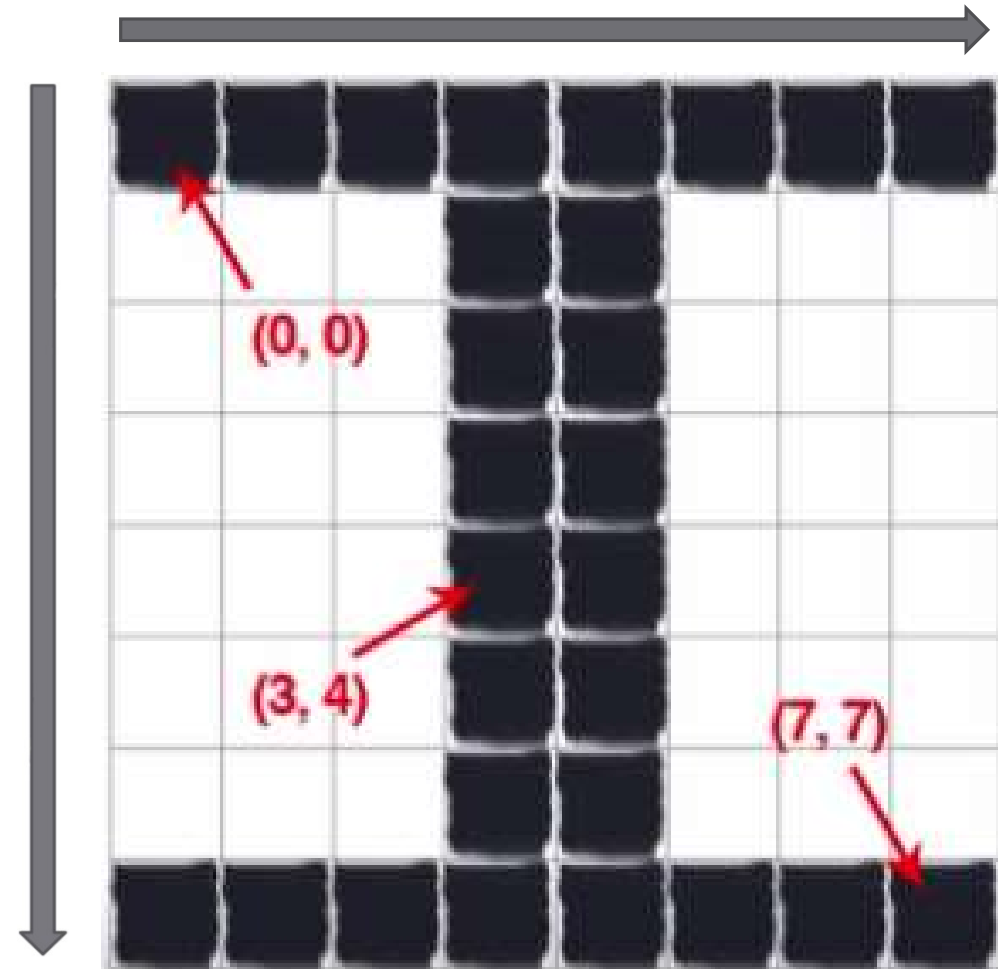
=

| | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 20 | 0 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 75 | 75 | 75 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 75 | 95 | 95 | 75 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 96 | 127 | 145 | 175 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 127 | 145 | 175 | 175 | 175 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 127 | 145 | 200 | 200 | 175 | 175 | 95 | 255 | 255 | 255 |
| 255 | 255 | 127 | 145 | 200 | 200 | 175 | 175 | 95 | 47 | 255 | 255 |
| 255 | 255 | 127 | 145 | 145 | 175 | 127 | 127 | 95 | 47 | 255 | 255 |
| 255 | 255 | 74 | 127 | 127 | 127 | 95 | 95 | 95 | 47 | 255 | 255 |
| 255 | 255 | 255 | 74 | 74 | 74 | 74 | 74 | 74 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |

OpenCV Fundamentals

- Image coordinate system

500



OpenCV Fundamentals

- Identify and modify pixel values
- Specify 4 values:
 - Starting and ending x-coordinates
 - Starting and ending y-coordinates

OpenCV Fundamentals

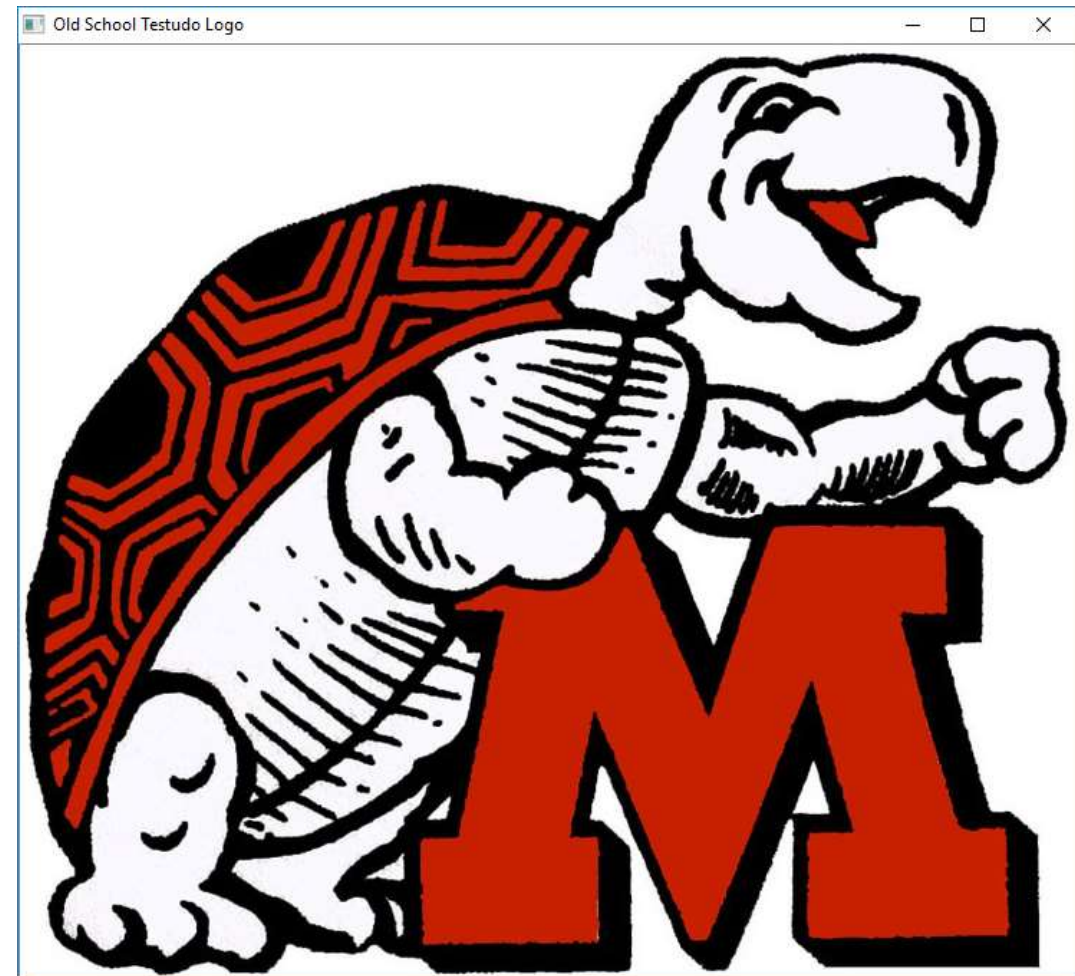
- Load image to screen

```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

cv2.imshow("Old School Testudo Logo", image)
cv2.waitKey(0)
```



OpenCV Fundamentals

- Images are represented as numpy arrays

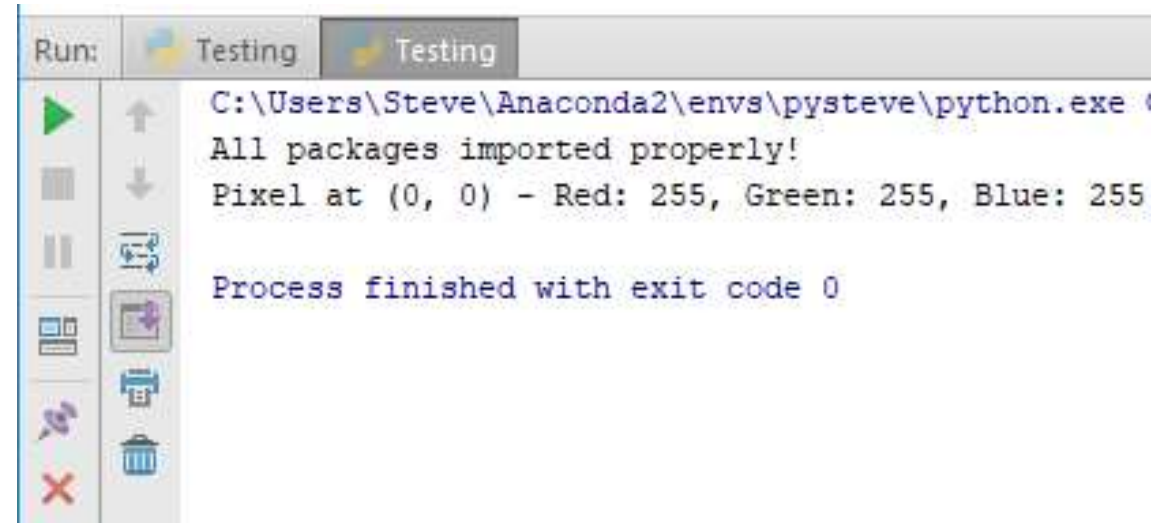
```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

cv2.imshow("Old School Testudo Logo", image)
cv2.waitKey(0)

(b, g, r) = image[0, 0]
print "Pixel at (0, 0) - Red: %d, Green: %d, Blue: %d" % (r, g, b)
```



***OpenCV represents images as numpy arrays in reverse order**

OpenCV Fundamentals

- Images are represented as numpy arrays

"The reason why the early developers at OpenCV chose BGR color format is probably that back then BGR color format was popular among camera manufacturers and software providers. E.g. in Windows, when specifying color value using COLORREF they use the BGR format 0x00bbggrr.

BGR was a choice made for historical reasons and now we have to live with it. In other words, BGR is the horse's ass in OpenCV."

***OpenCV represents images as numpy arrays in reverse order**

OpenCV Fundamentals

- Modify pixel color

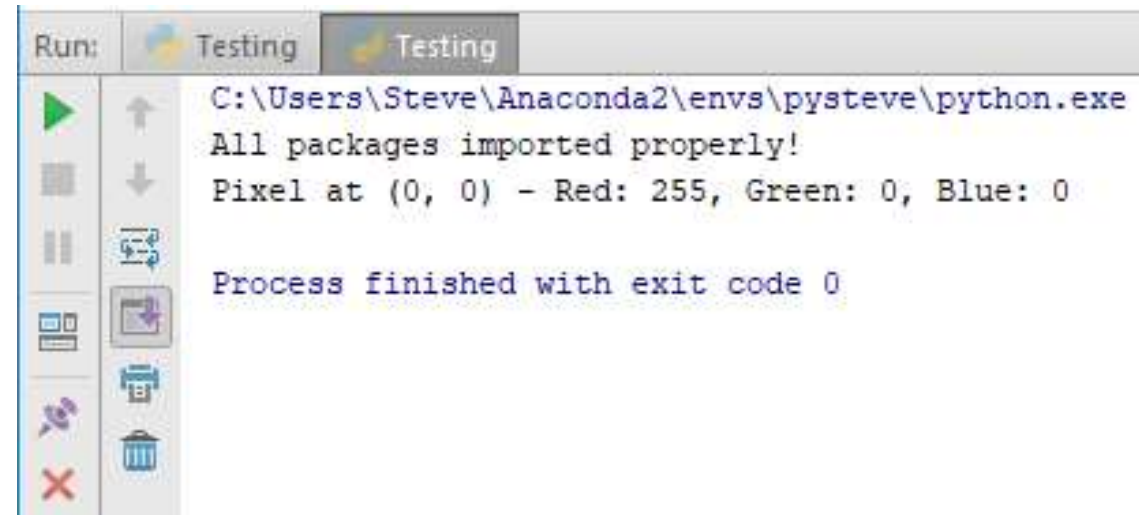
```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

cv2.imshow("Old School Testudo Logo", image)
cv2.waitKey(0)

image[0, 0] = (0, 0, 255)
(b, g, r) = image[0, 0]
print "Pixel at (0, 0) - Red: %d, Green: %d, Blue: %d" % (r, g, b)
```



OpenCV Fundamentals

- Image slicing

```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

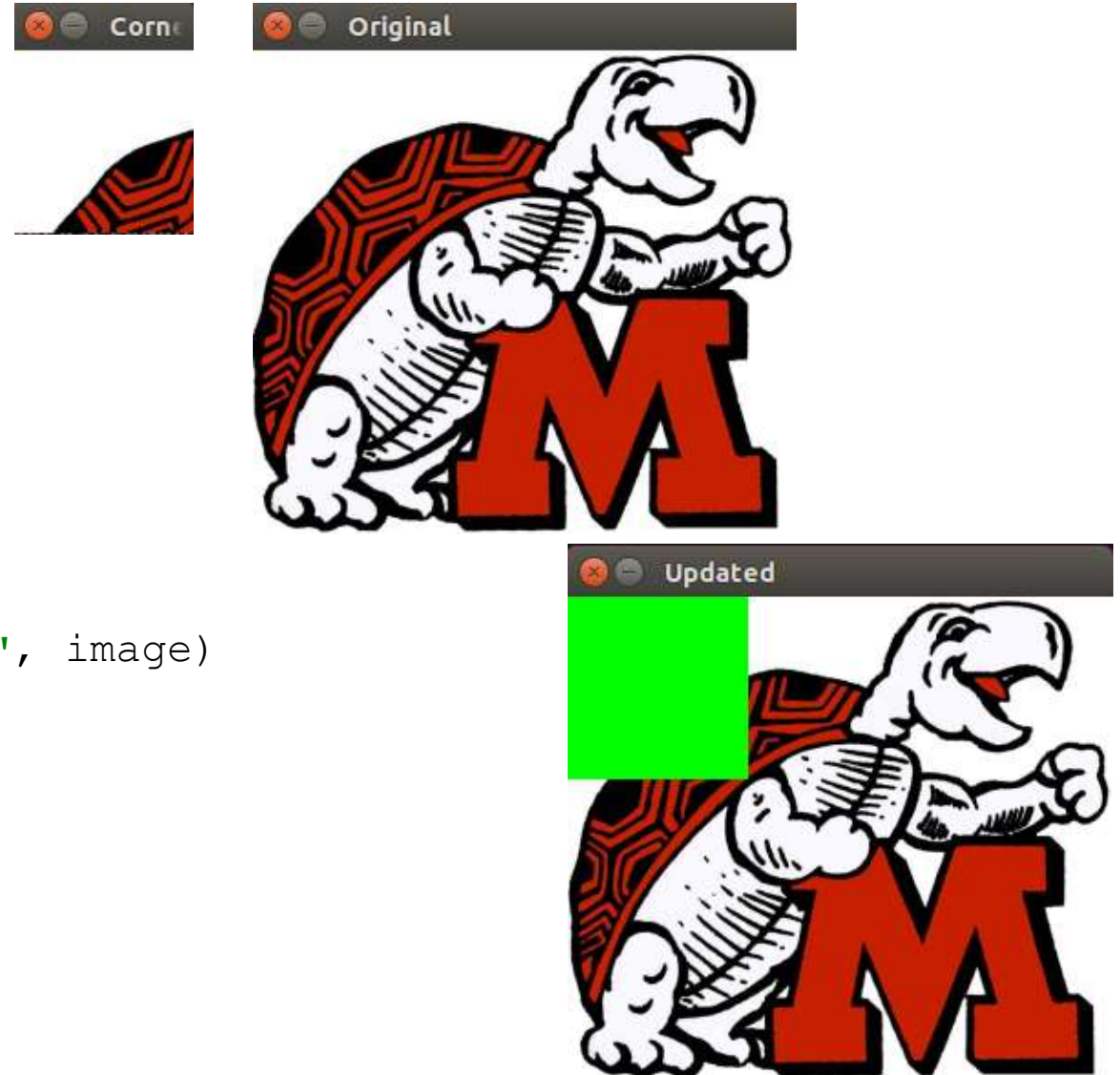
image = imutils.resize(image, width=400)

cv2.imshow("Old School Testudo Logo: Resized", image)

corner = image[0:100, 0:100]
cv2.imshow("Corner", corner)

image[0:100, 0:100] = (0, 255, 0)

cv2.imshow("Updated", image)
cv2.waitKey(0)
```



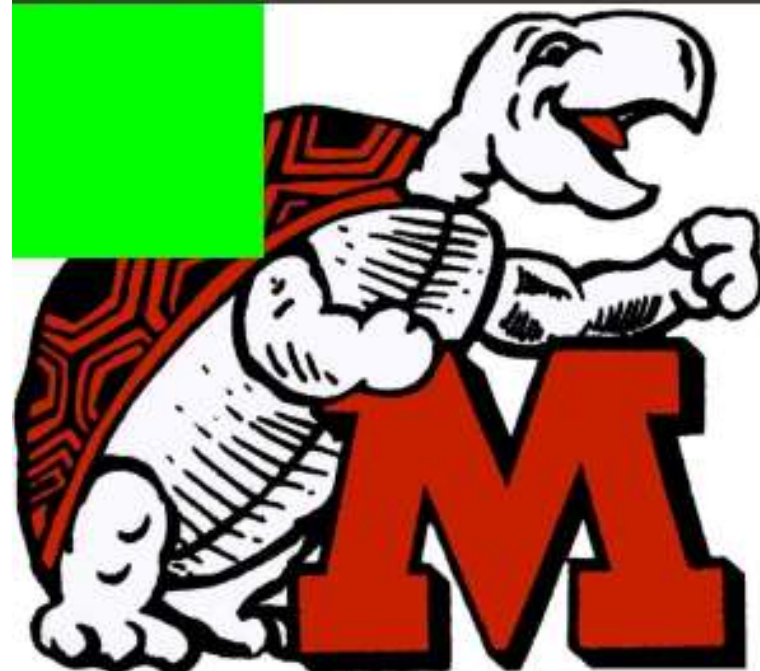
Original



Corn



Updated



OpenCV Fundamentals

- Image filtering/blurring
- Average
- Gaussian
- Median
- Preserve edges using bilateral filtering

OpenCV Fundamentals

- Modify pixels in an image based on a function of a local neighborhood of each pixel

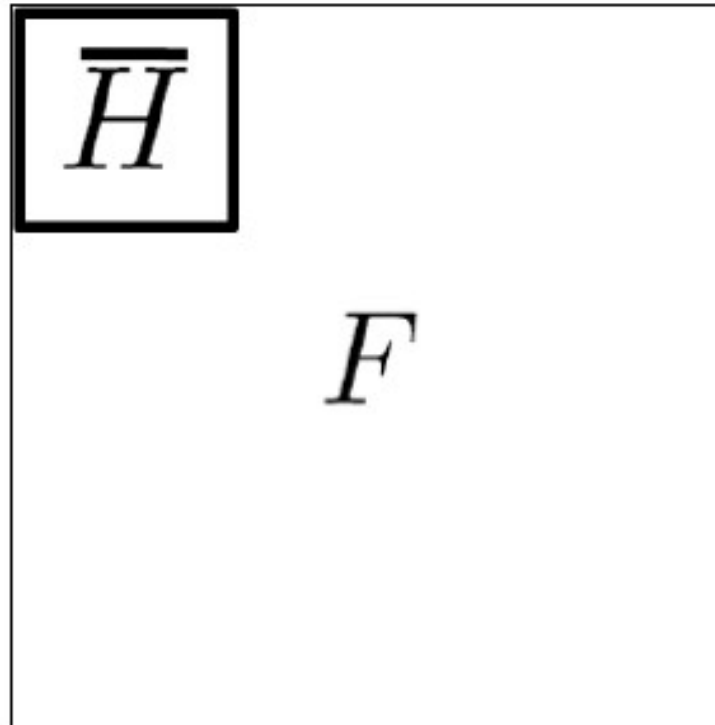


OpenCV Fundamentals

- Convolution

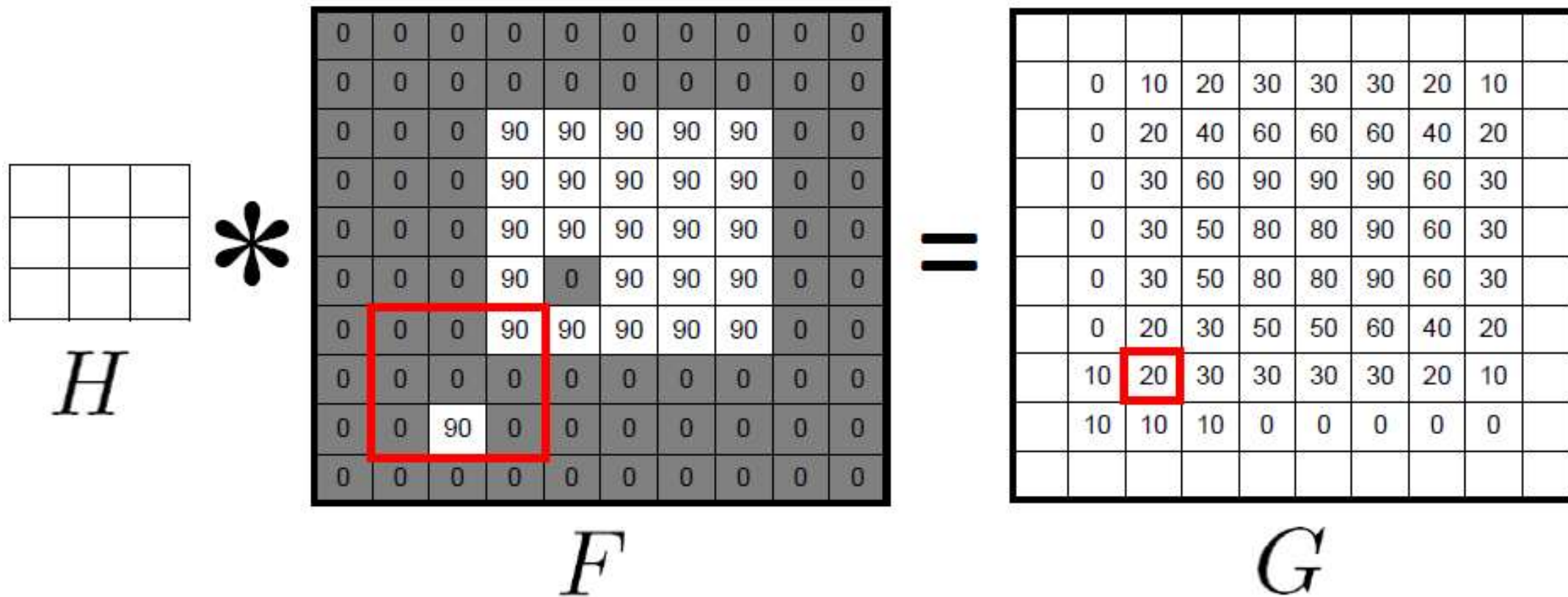


$$K = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



OpenCV Fundamentals

- Average blurring
- Fastest in terms of processing



OpenCV Fundamentals

- Average blurring

```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

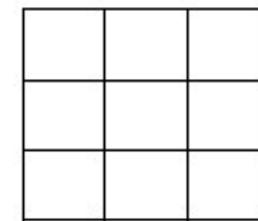
image = cv2.imread("testudo.jpg")

image = imutils.resize(image, width=400)

cv2.imshow("Old School Testudo Logo: Resized", image)

blurred = np.hstack([
    cv2.blur(image, (3,3)),
    cv2.blur(image, (5,5)),
    cv2.blur(image, (7,7))])
cv2.imshow("Average Blurring", blurred)

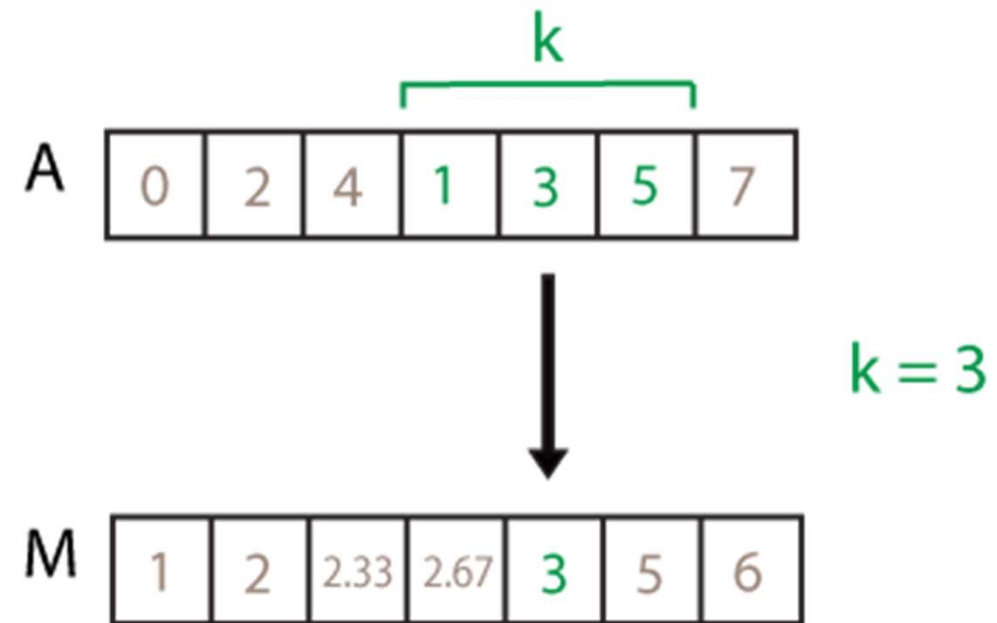
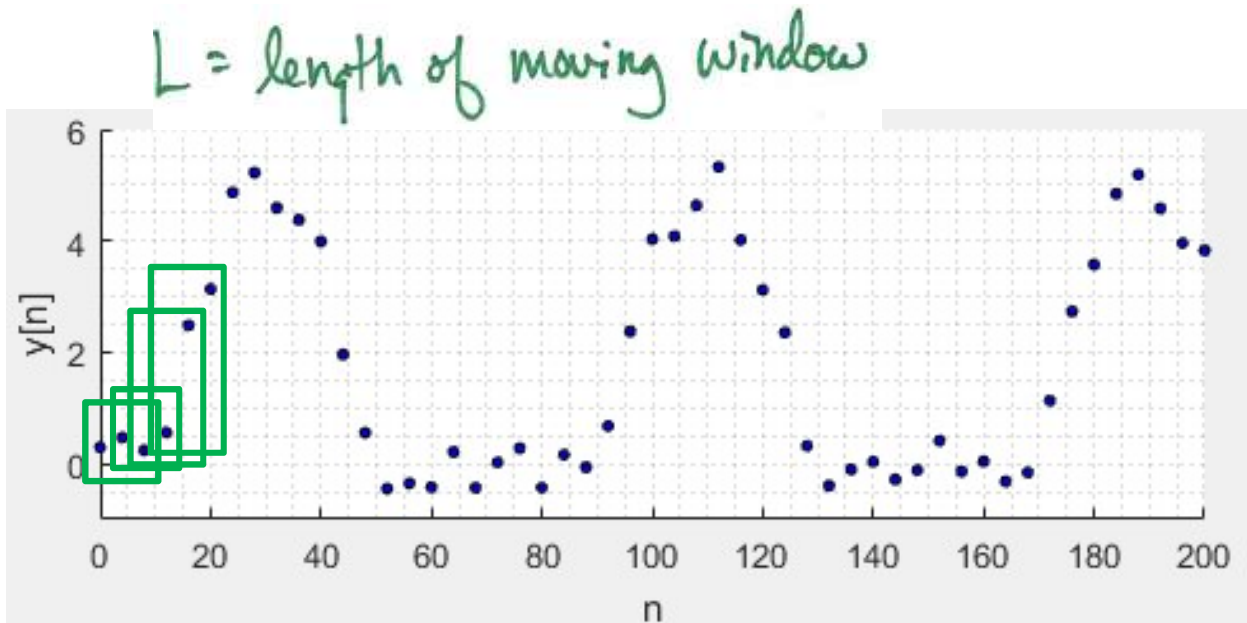
cv2.waitKey(0)
```



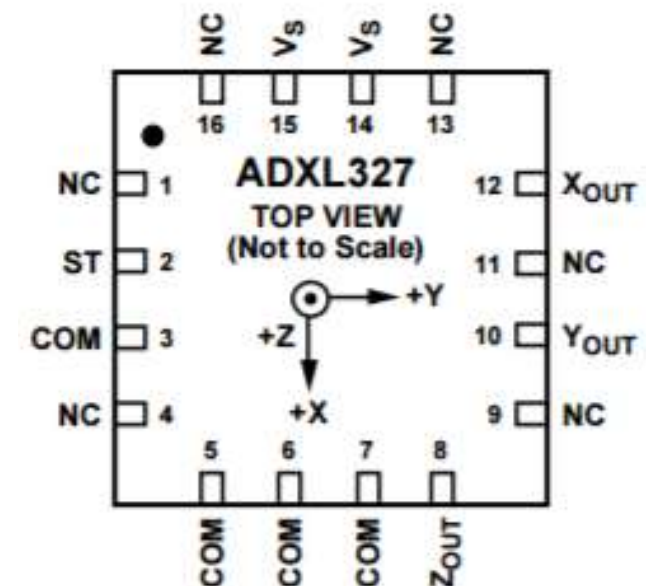
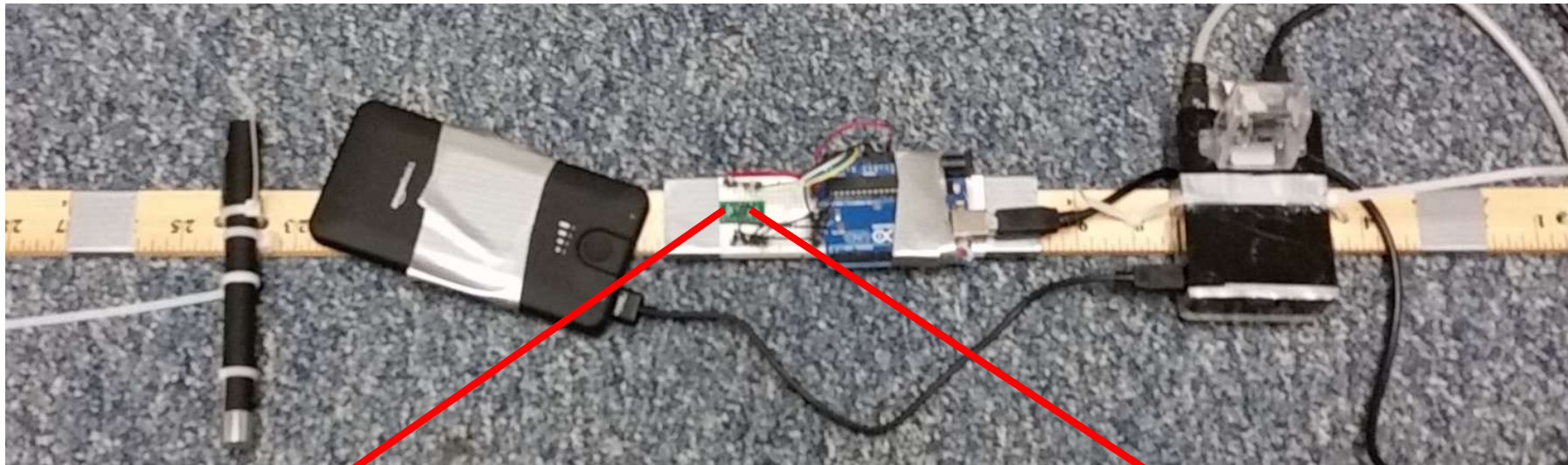
H



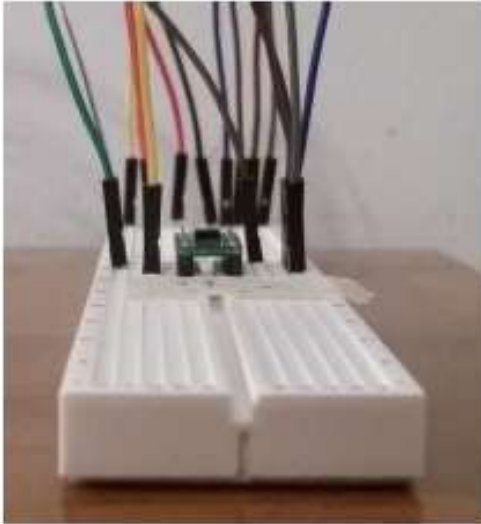
Digital filtering / moving average



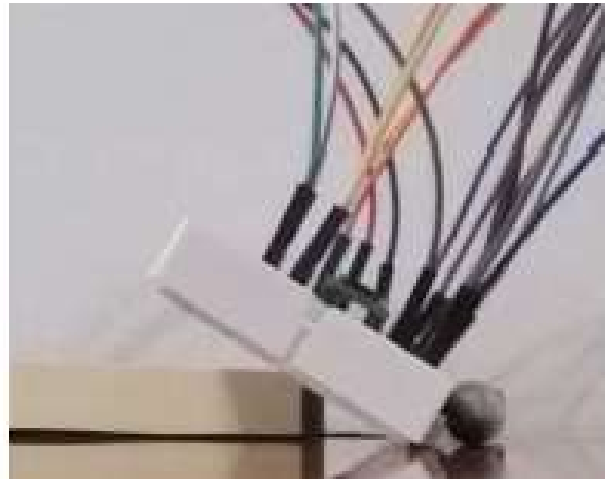
Moving average



Moving average



```
x = 332; Xangle = 0;  
x = 333; Xangle = 0;  
x = 333; Xangle = 1;  
x = 333; Xangle = 0;  
x = 332; Xangle = 0;  
x = 332; Xangle = 0;  
x = 333; Xangle = 0;
```

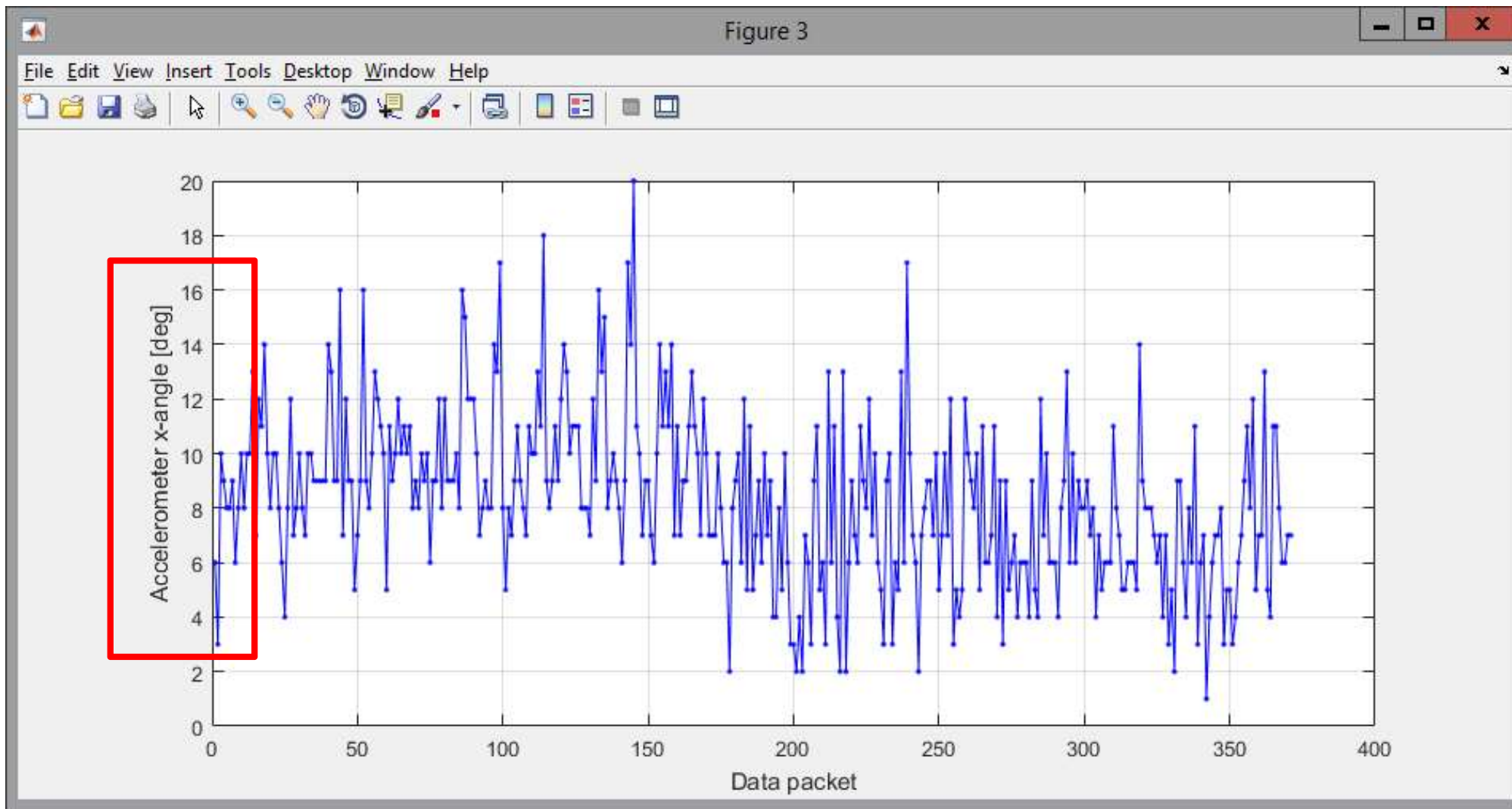


```
x = 285; Xangle = -44;  
x = 287; Xangle = -44;  
x = 286; Xangle = -45;  
x = 286; Xangle = -45;  
x = 287; Xangle = -46;  
x = 286; Xangle = -45;
```

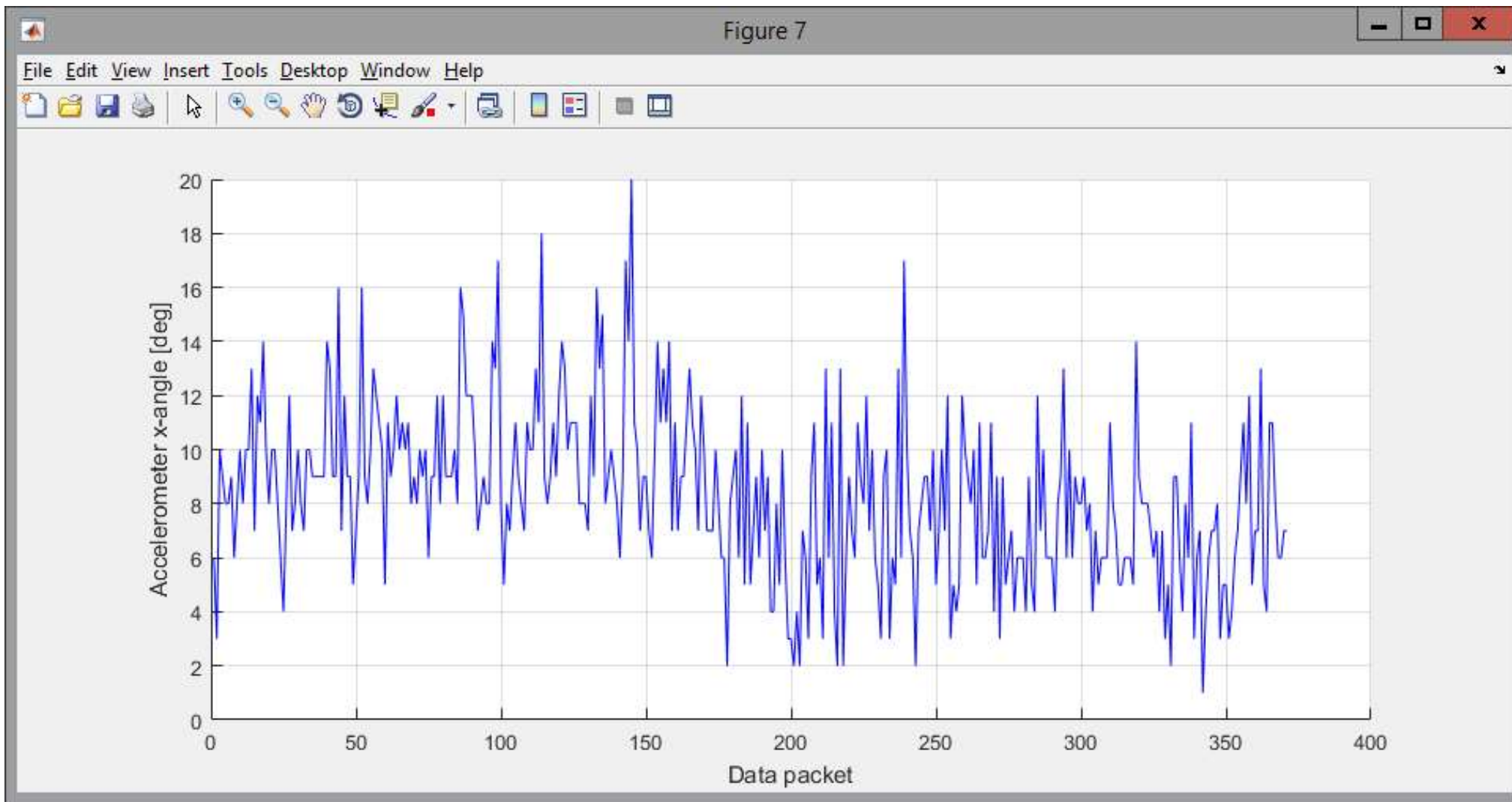


```
x = 380; Xangle = 44;  
x = 379; Xangle = 43;  
x = 380; Xangle = 45;  
x = 379; Xangle = 47;  
x = 380; Xangle = 45;  
x = 379; Xangle = 44;
```

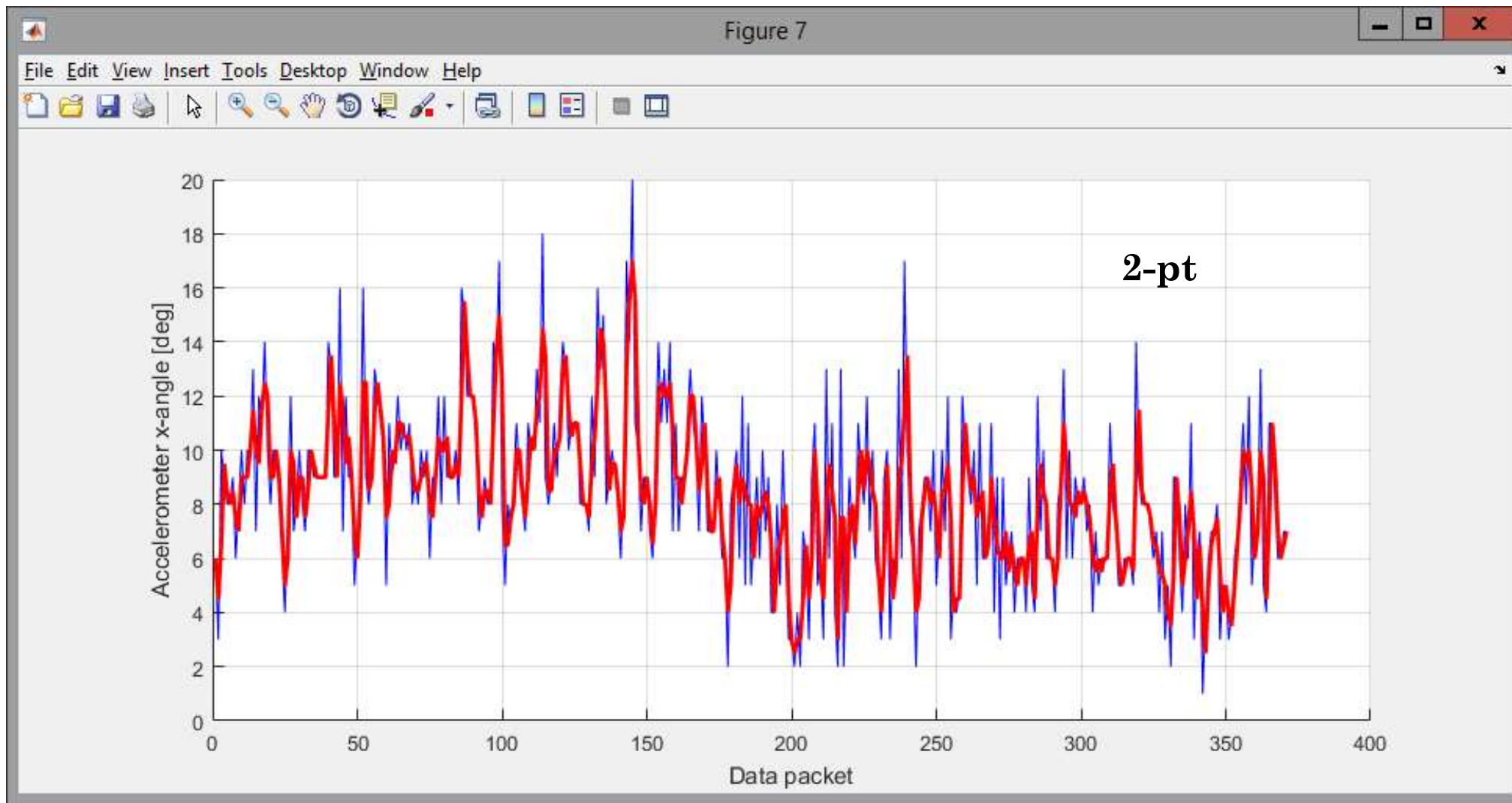

Moving average



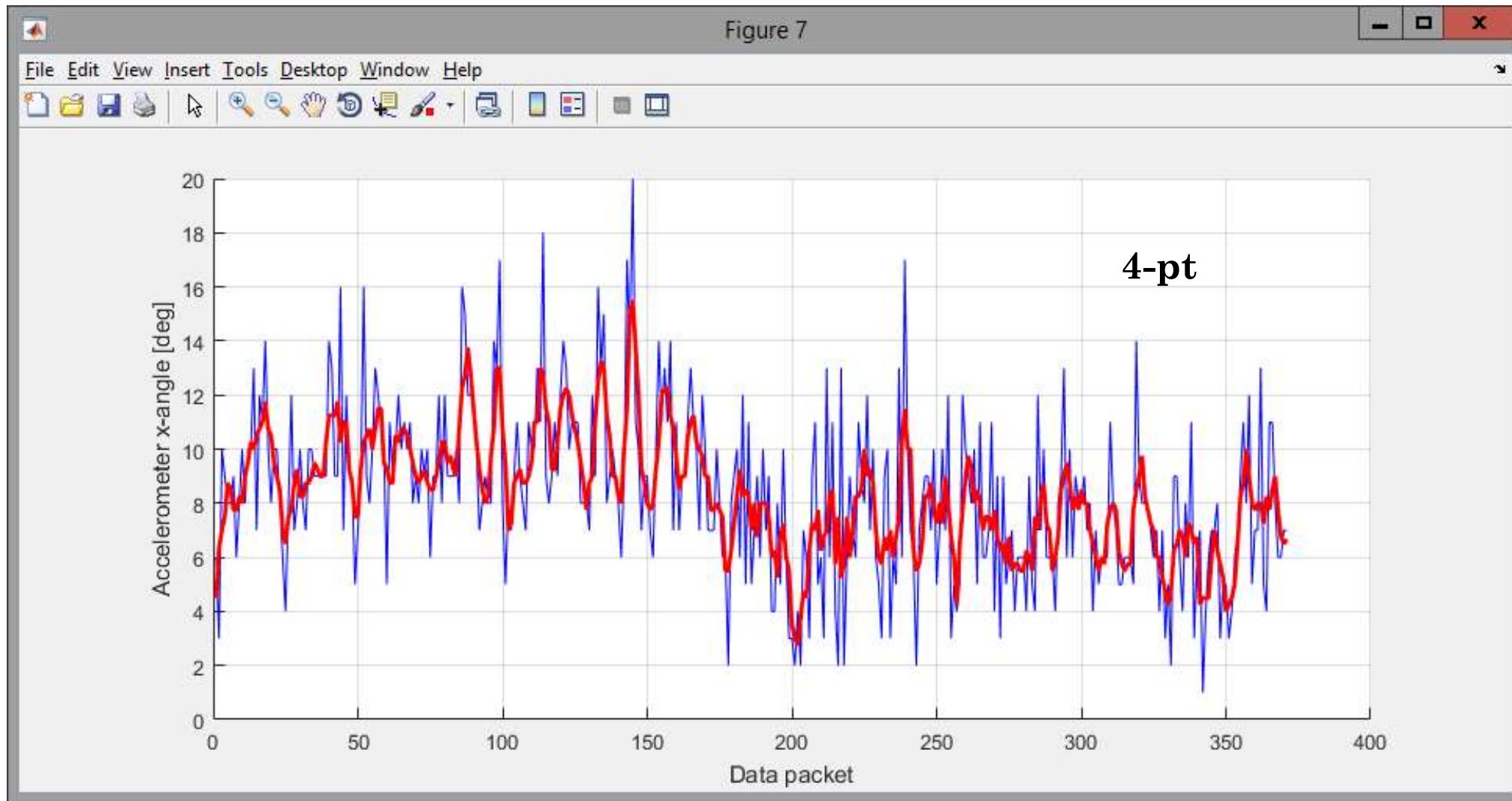
Moving average



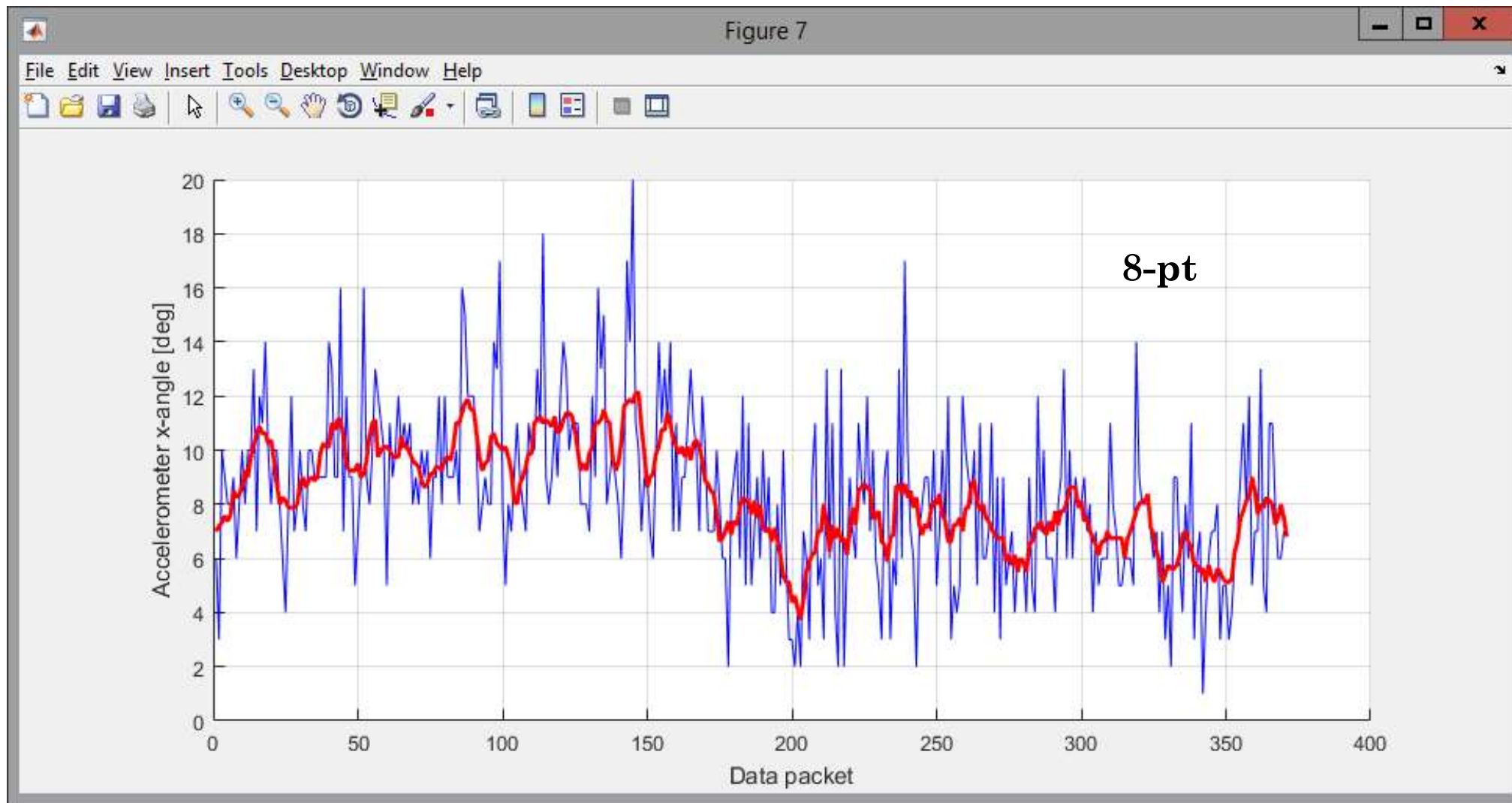
Moving average



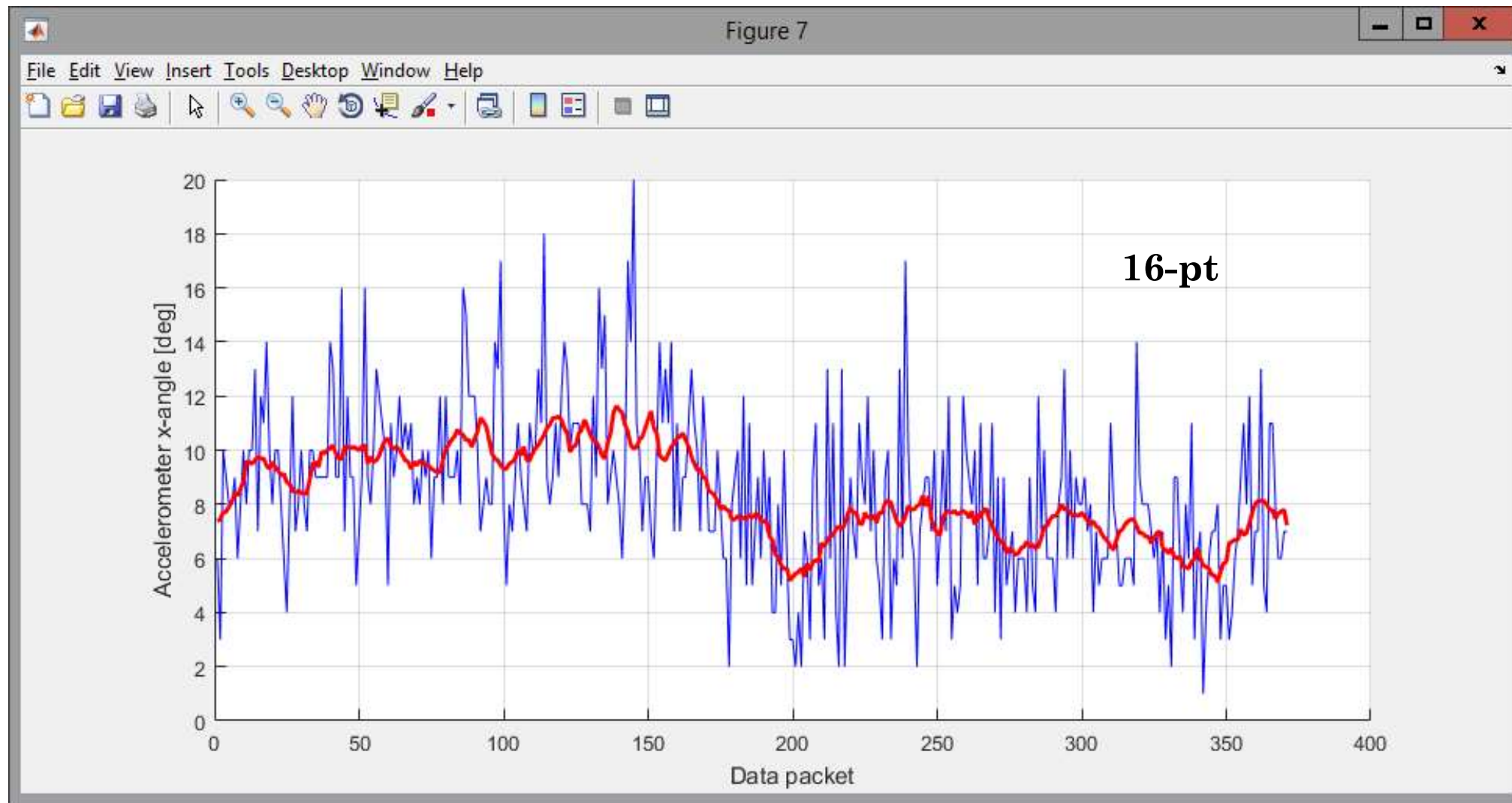
Moving average



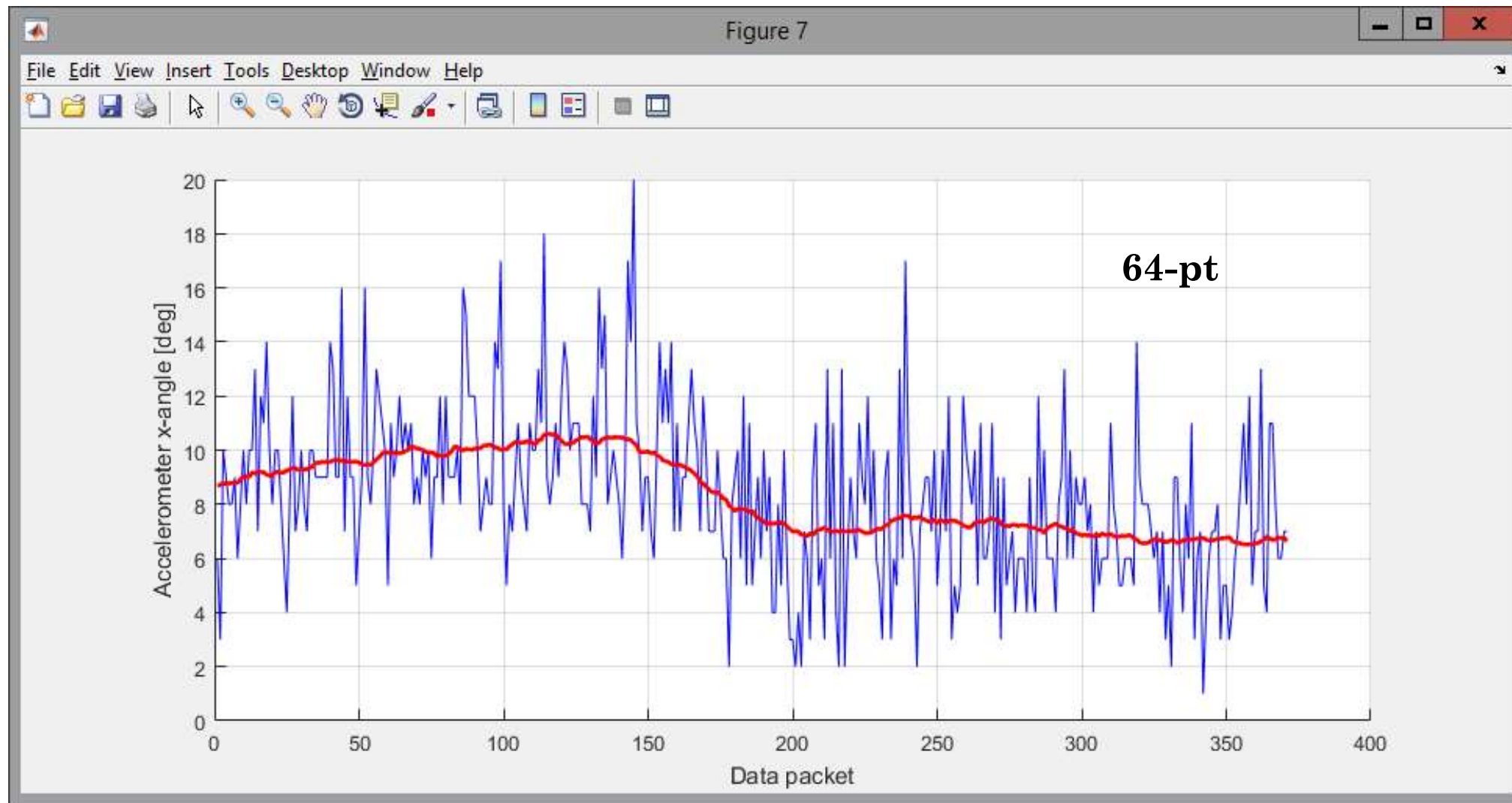
Moving average



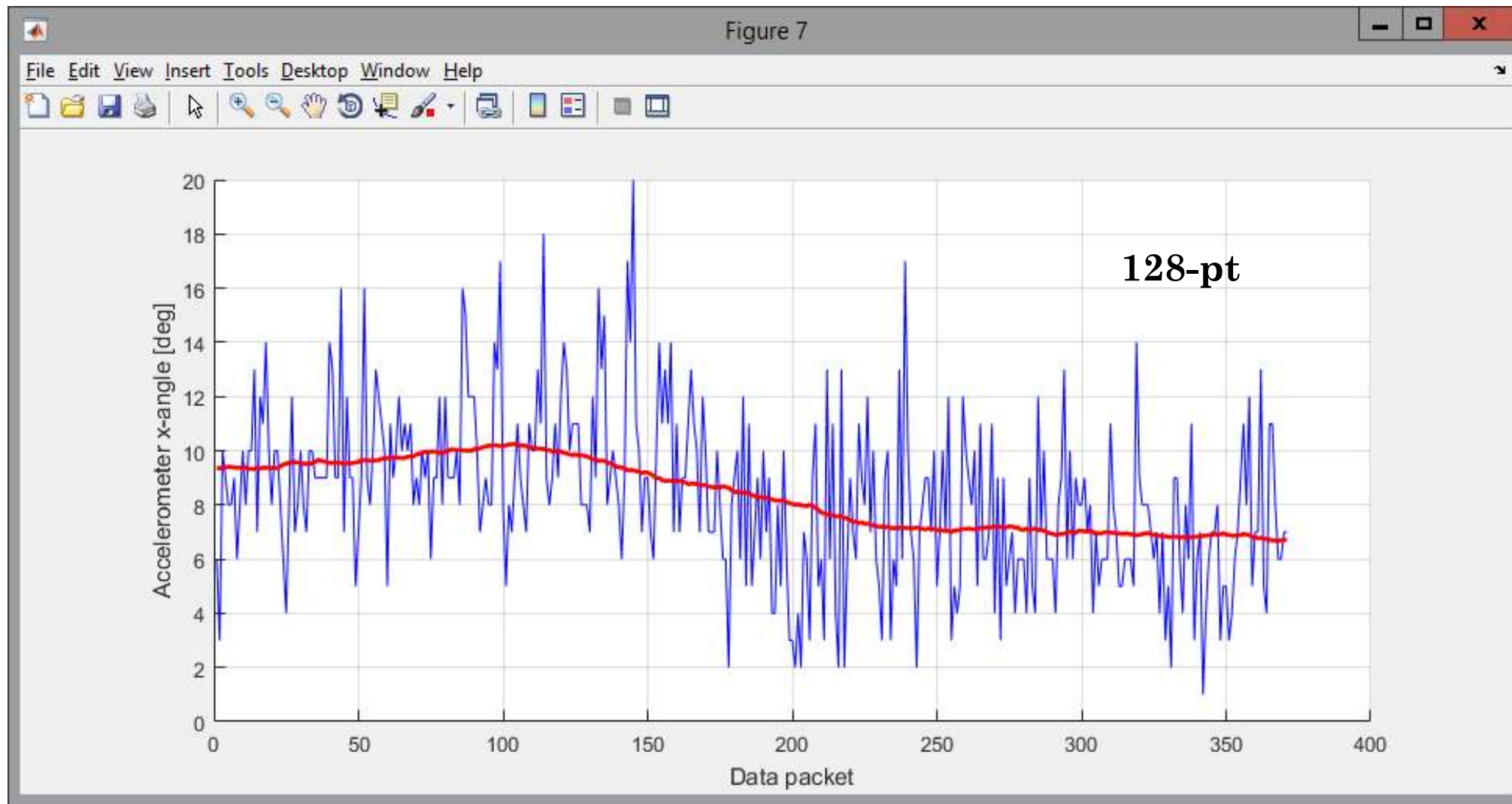
Moving average



Moving average

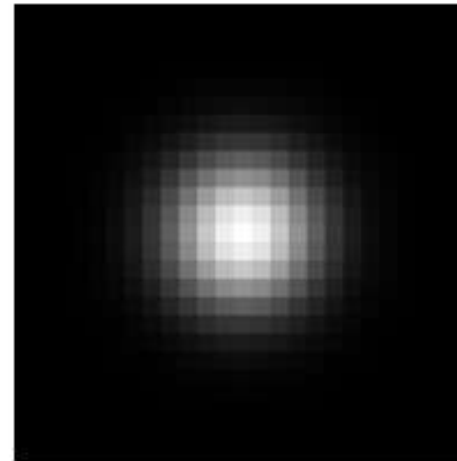
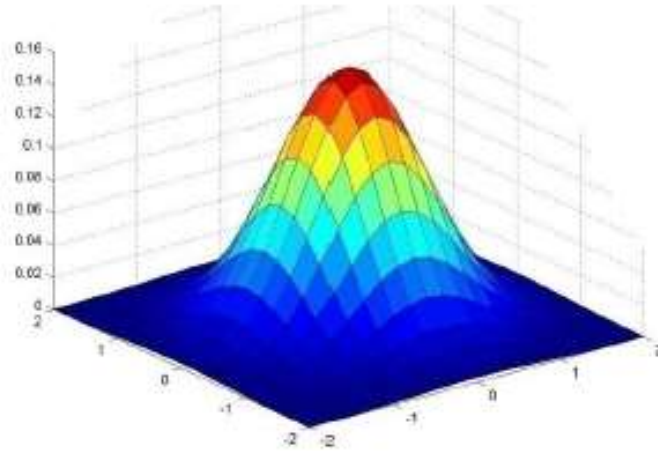


Moving average



OpenCV Fundamentals

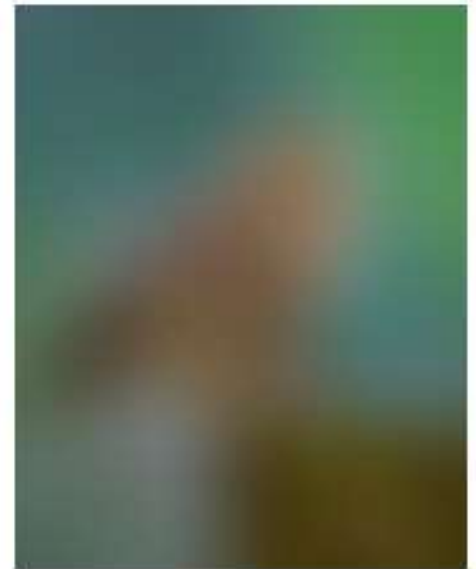
- Gaussian blurring



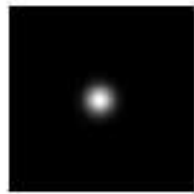
$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

OpenCV Fundamentals

- Gaussian blurring



$\sigma = 1$ pixel



$\sigma = 5$ pixels



$\sigma = 10$ pixels



$\sigma = 30$ pixels

OpenCV Fundamentals

- Gaussian blurring

```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")

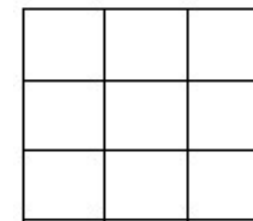
image = imutils.resize(image, width=400)

cv2.imshow("Old School Testudo Logo: Resized", image)

blurred = np.hstack([
    cv2.GaussianBlur(image, (3,3), 0),
    cv2.GaussianBlur(image, (5,5), 0),
    cv2.GaussianBlur(image, (7,7), 0)])
cv2.imshow("Gaussian Blurring", blurred)

cv2.waitKey(0)
```

Automatically compute
based on kernel
size



H



OpenCV Fundamentals

- Median blurring
- Replace central pixel with median of neighborhood
- Most effective at removing salt & pepper noise

<https://snag.gy/bv87f.jpg>

Original Color Image



Color Image with Salt and Pepper Noise



Restored Image



OpenCV Fundamentals

- Median blurring

```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

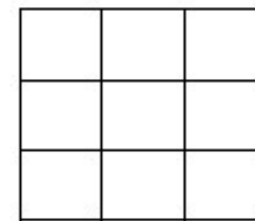
image = cv2.imread("testudo.jpg")

image = imutils.resize(image, width=400)

cv2.imshow("Old School Testudo Logo: Resized", image)

blurred = np.hstack([
    cv2.medianBlur(image, 3),
    cv2.medianBlur(image, 5),
    cv2.medianBlur(image, 7)])
cv2.imshow("Median Blurring", blurred)

cv2.waitKey(0)
```



H



OpenCV Fundamentals

- Bilateral filter
 - Reduce noise while maintaining edges
- Two Gaussians
 - First considers pixels close together in (x, y)
 - Second ensures only pixels with similar intensity are included in computation of blur
- Computationally demanding

OpenCV Fundamentals

- Bilateral filter

```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

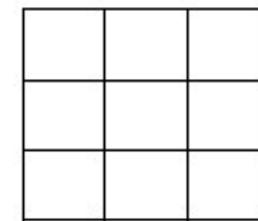
image = cv2.imread("testudo.jpg")

image = imutils.resize(image, width=400)

cv2.imshow("Old School Testudo Logo: Resized", image)

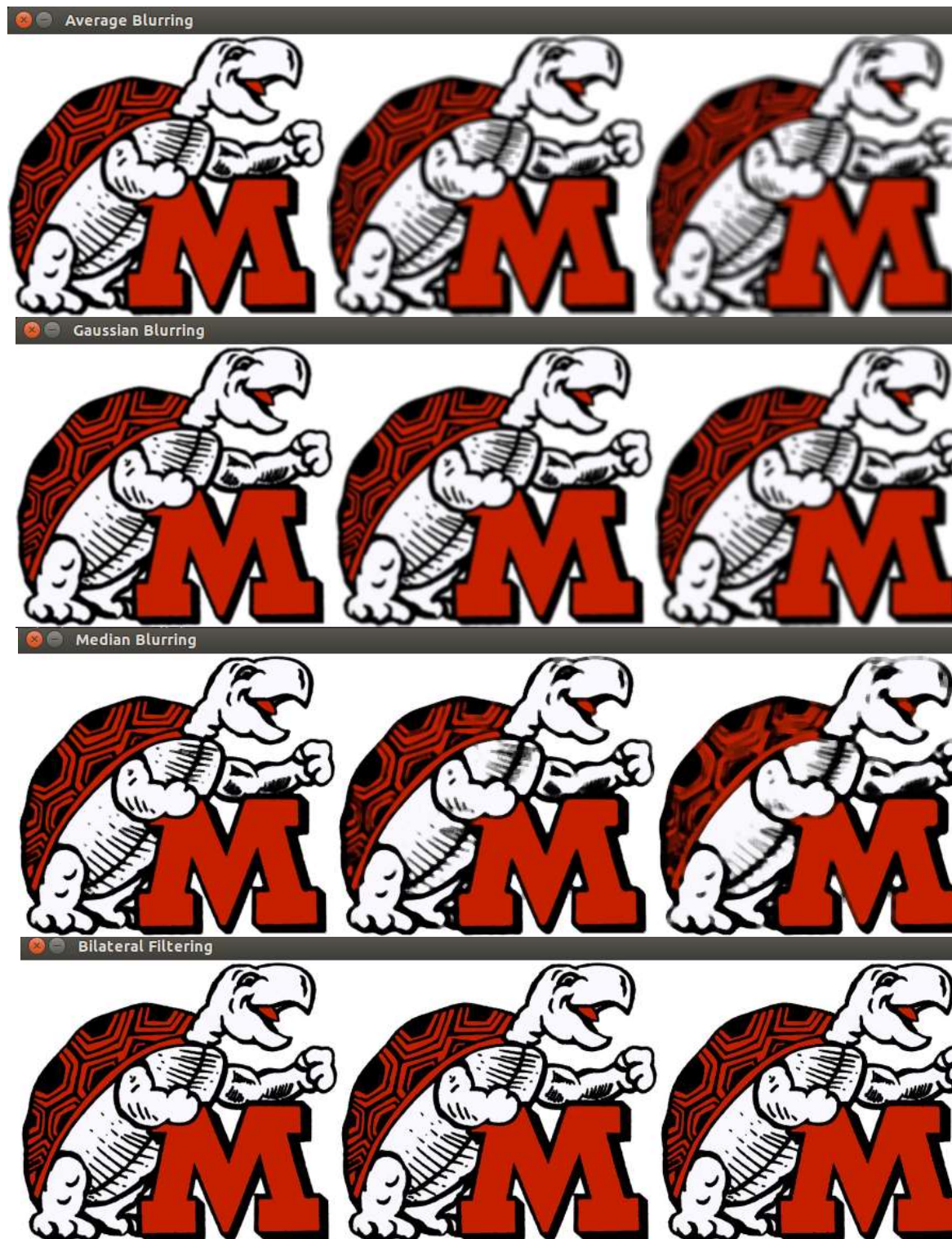
blurred = np.hstack([
    cv2.bilateralFilter(image, 5, 21, 21),
    cv2.bilateralFilter(image, 7, 31, 31),
    cv2.bilateralFilter(image, 9, 41, 41)])
cv2.imshow("Bilater Filtering", blurred)

cv2.waitKey(0)
```



H





OpenCV Fundamentals

- Drawing lines
- Rectangles
- Circles

OpenCV Fundamentals

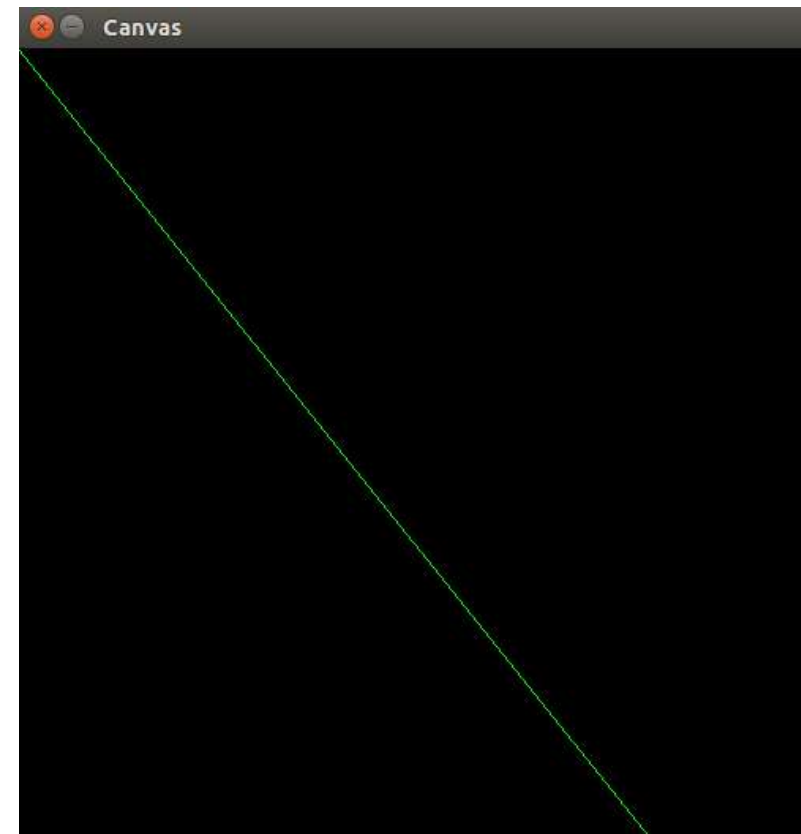
- Import packages and define a blank canvas (image)
- Draw a line atop canvas

```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

canvas = np.zeros((500, 500, 3), dtype="uint8")

green = (0, 255, 0)
cv2.line(canvas, (0,0), (400, 500), green)
cv2.imshow("Canvas", canvas)
cv2.waitKey(0)
```



OpenCV Fundamentals

- Green and red lines defined by starting/ending coordinates

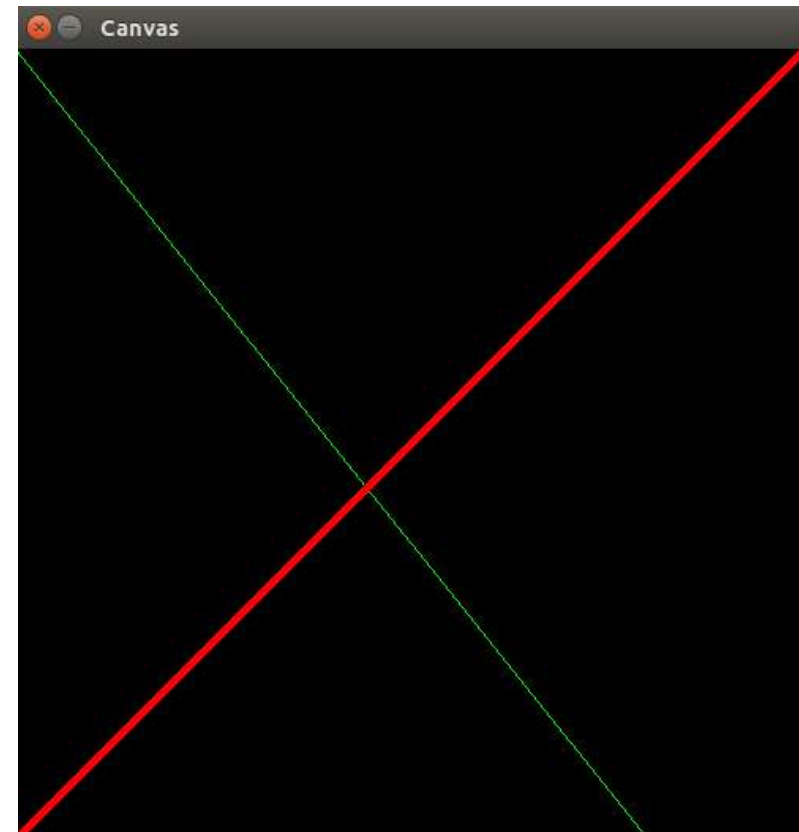
```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

canvas = np.zeros((500, 500, 3), dtype="uint8")

green = (0, 255, 0)
cv2.line(canvas, (0,0), (400, 500), green)

red = (0, 0, 255)
cv2.line(canvas, (500, 0), (0, 500), red, 3)
cv2.imshow("Canvas", canvas)
cv2.waitKey(0)
```



OpenCV Fundamentals

- Rectangle: green outline, no fill

```
import numpy as np
import cv2
import imutils

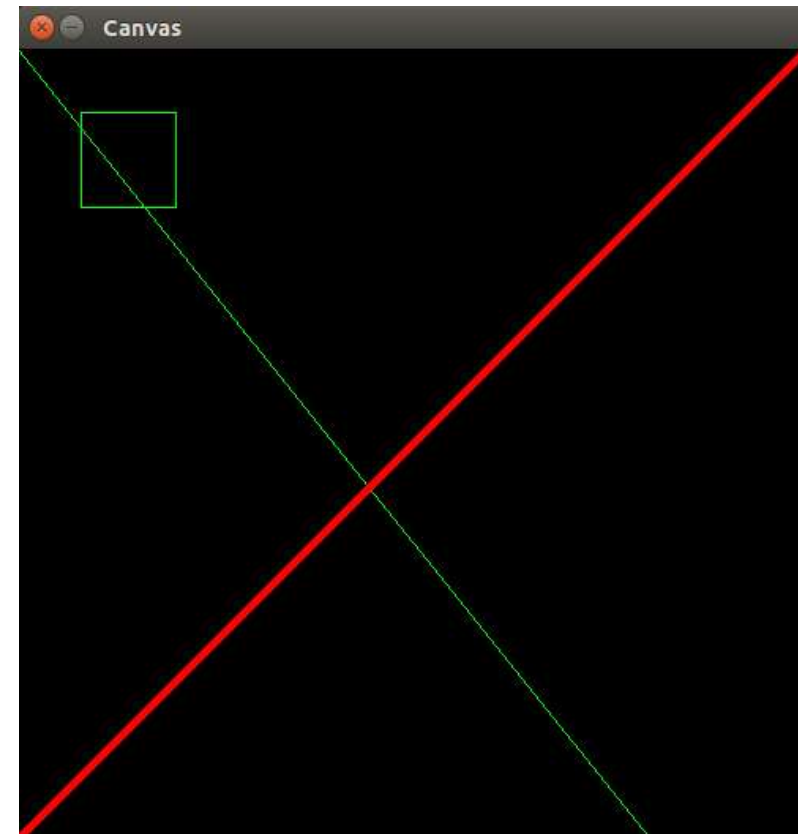
print "All packages imported properly!"

canvas = np.zeros((500, 500, 3), dtype="uint8")

green = (0, 255, 0)
cv2.line(canvas, (0,0), (400, 500), green)

red = (0, 0, 255)
cv2.line(canvas, (500, 0), (0, 500), red, 3)

cv2.rectangle(canvas, (40, 50), (100, 100), green)
cv2.imshow("Canvas", canvas)
cv2.waitKey(0)
```



OpenCV Fundamentals

- Red rectangle no fill and blue filled rectangle

```
import numpy as np
import cv2
import imutils

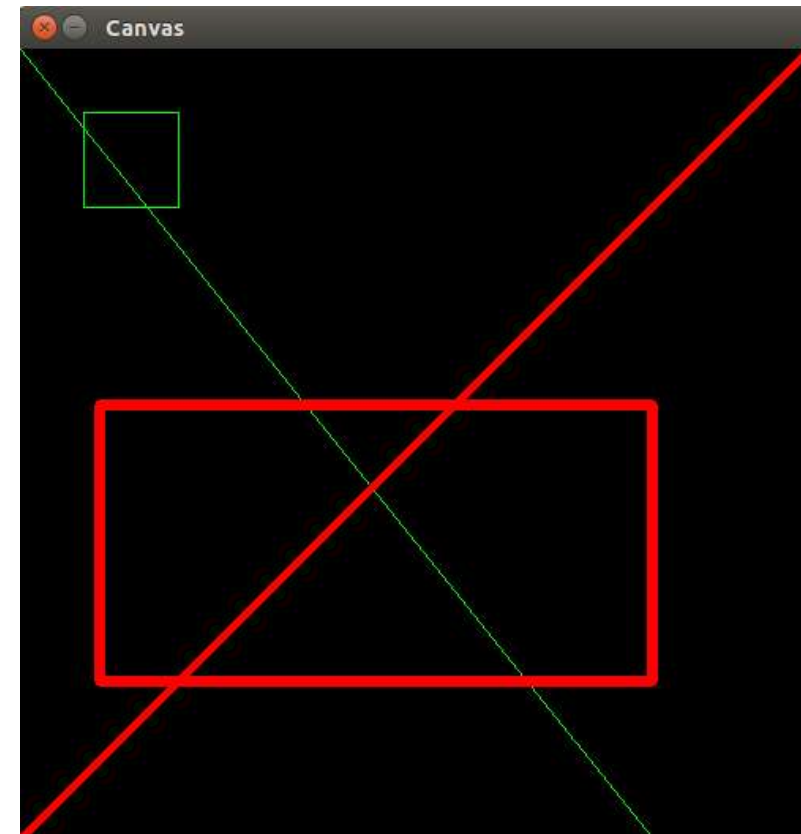
print "All packages imported properly!"

canvas = np.zeros((500, 500, 3), dtype="uint8")

green = (0, 255, 0)
cv2.line(canvas, (0,0), (400, 500), green)

red = (0, 0, 255)
cv2.line(canvas, (500, 0), (0, 500), red, 3)

cv2.rectangle(canvas, (40, 50), (100, 100), green)
cv2.rectangle(canvas, (50, 400), (400, 225), red, 5)
cv2.imshow("Canvas", canvas)
cv2.waitKey(0)
```



OpenCV Fundamentals

- Blue filled rectangle

```
import numpy as np
import cv2
import imutils

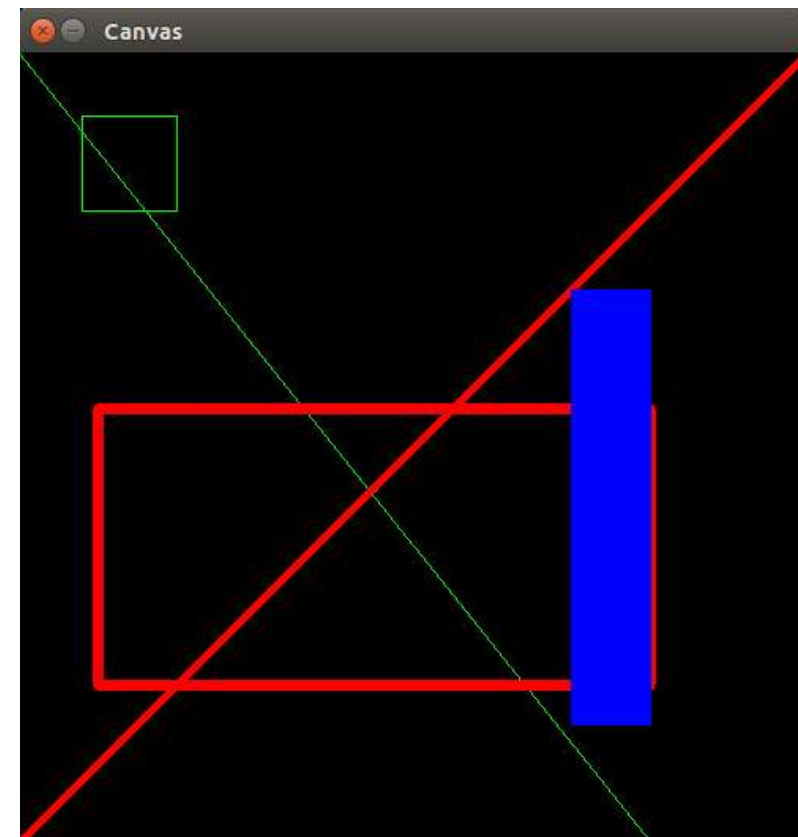
print "All packages imported properly!"

canvas = np.zeros((500, 500, 3), dtype="uint8")

green = (0, 255, 0)
cv2.line(canvas, (0,0), (400, 500), green)

red = (0, 0, 255)
cv2.line(canvas, (500, 0), (0, 500), red, 3)

cv2.rectangle(canvas, (40, 50), (100, 100), green)
cv2.rectangle(canvas, (50, 400), (400, 225), red, 5)
cv2.rectangle(canvas, (350, 150), (400, 425), (255, 0, 0), -1)
cv2.imshow("Canvas", canvas)
cv2.waitKey(0)
```



OpenCV Fundamentals

- Concentric circles

```
import numpy as np
import cv2
import imutils

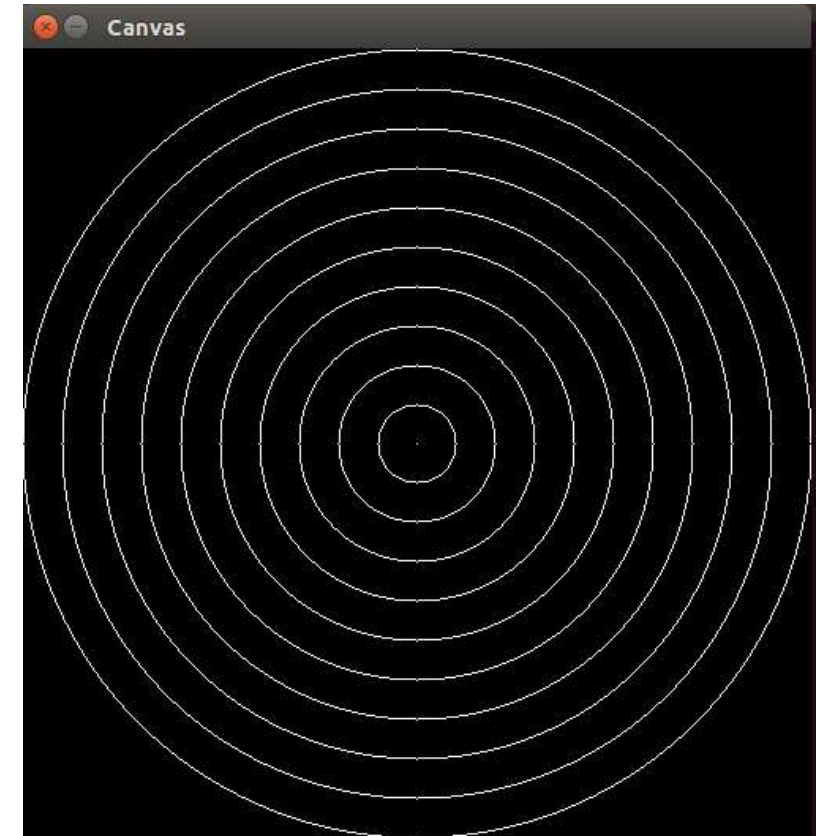
print "All packages imported properly!"

canvas = np.zeros((500, 500, 3), dtype="uint8")

(centerX, centerY) = (canvas.shape[1]/2, canvas.shape[0]/2)

white = (255, 255, 255)

for r in xrange(0, 275, 25):
    cv2.circle(canvas, (centerX, centerY), r, white)
    cv2.imshow("Concentric Circles", canvas)
    cv2.waitKey(0)
```



OpenCV Fundamentals

- Overlay text on image

```
import numpy as np
import cv2
import imutils

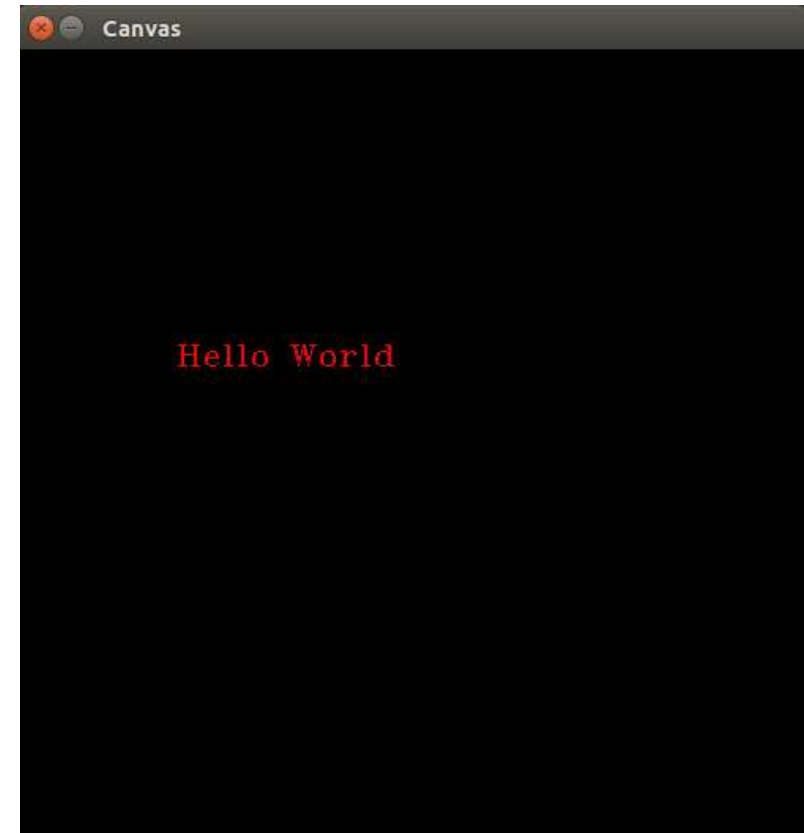
print "All packages imported properly!"

canvas = np.zeros((500, 500, 3), dtype="uint8")

font = cv2.FONT_HERSHEY_COMPLEX_SMALL

red = (0, 0, 255)
cv2.putText(canvas, 'Hello World', (100, 200), font, 1, red, 1)

cv2.imshow("Canvas", canvas)
cv2.waitKey(0)
```



OpenCV Fundamentals

- Image transformations / flipping
- Masking

OpenCV Fundamentals

- Flip image

```
import numpy as np
import cv2
import imutils

image = cv2.imread("testudo.jpg")
image = imutils.resize(image, width=400)

cv2.imshow("Original", image)

flipped = cv2.flip(image, 1)
cv2.imshow("Flipped Horizontally", flipped)

flipped = cv2.flip(image, 0)
cv2.imshow("Flipped Vertically", flipped)

flipped = cv2.flip(image, -1)
cv2.imshow("Flipped Horizontally & Vertically", flipped)

cv2.waitKey(0)
```



OpenCV Fundamentals

- Masking

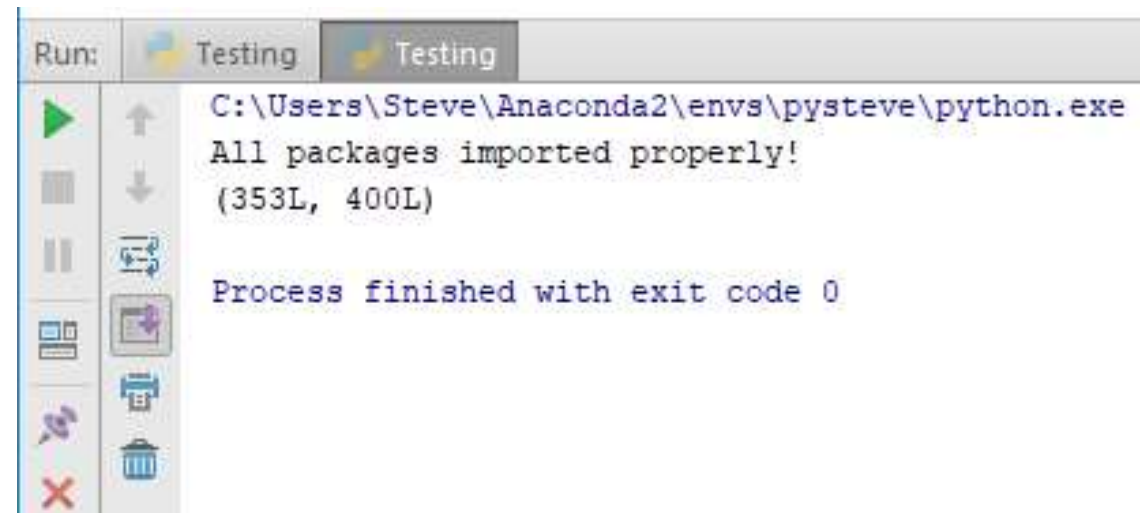
```
import numpy as np
import cv2
import imutils

print "All packages imported properly!"

image = cv2.imread("testudo.jpg")
image = imutils.resize(image, width=400)

cv2.imshow("Original", image)

print image.shape[:2]
```



OpenCV Fundamentals

- Rectangular mask

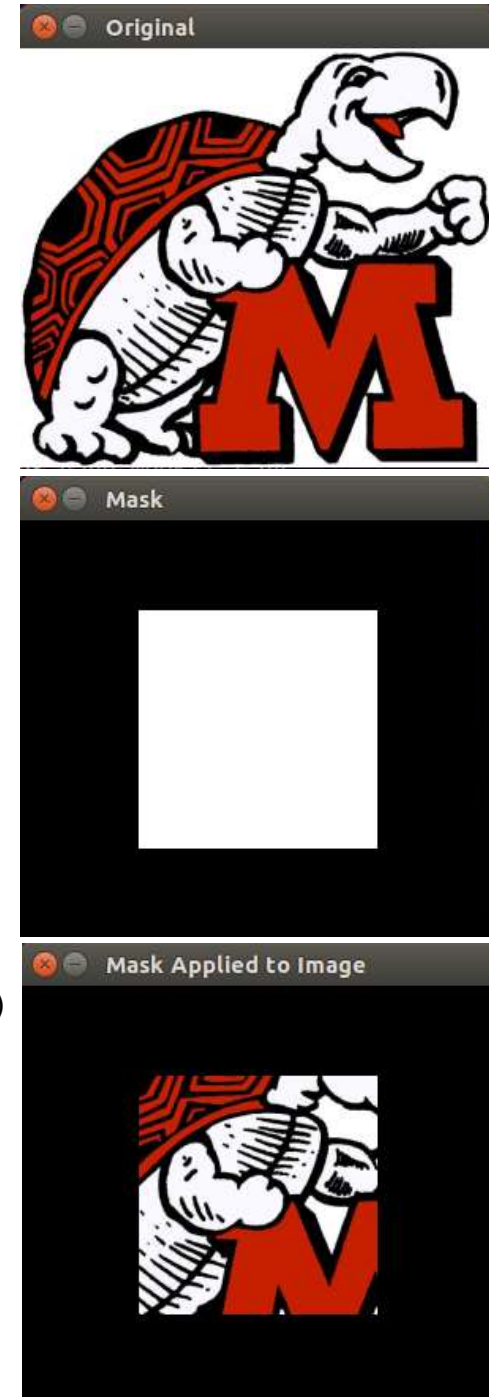
```
import numpy as np
import cv2
import imutils

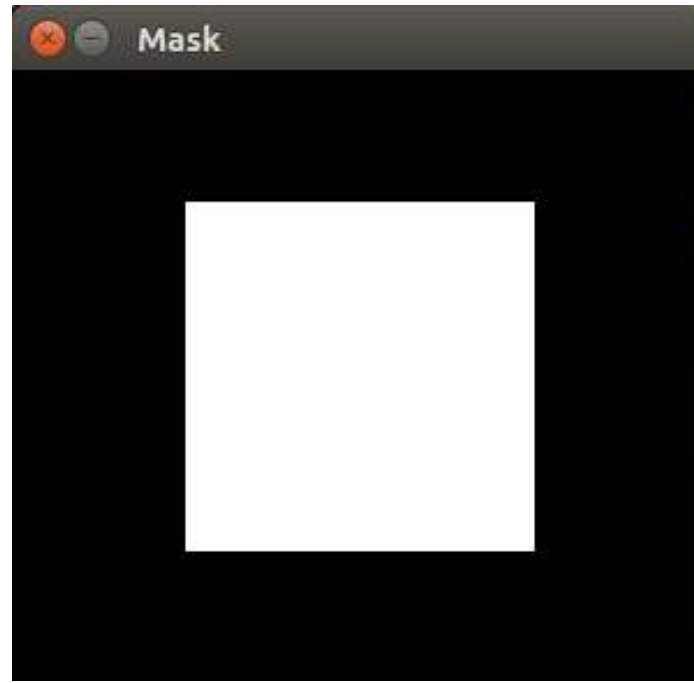
image = cv2.imread("testudo.jpg")
image = imutils.resize(image, width=400)
cv2.imshow("Original", image)

mask = np.zeros(image.shape[:2], dtype = "uint8")
(cX, cY) = (image.shape[1]/2, image.shape[0]/2)
cv2.rectangle(mask, (cX - 75, cY - 75), (cX + 75, cY + 75), 255, -1)
cv2.imshow("Mask", mask)

masked = cv2.bitwise_and(image, image, mask=mask)
cv2.imshow("Mask Applied to Image", masked)

cv2.waitKey(0)
```





AND logic gate

Gate



Truth Table

| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Notation

$$Z = AB$$
$$= A \cdot B$$

↑
"logic
multiplication"

- Output is 1 if **BOTH** inputs are 1

OpenCV Fundamentals

- Circular mask

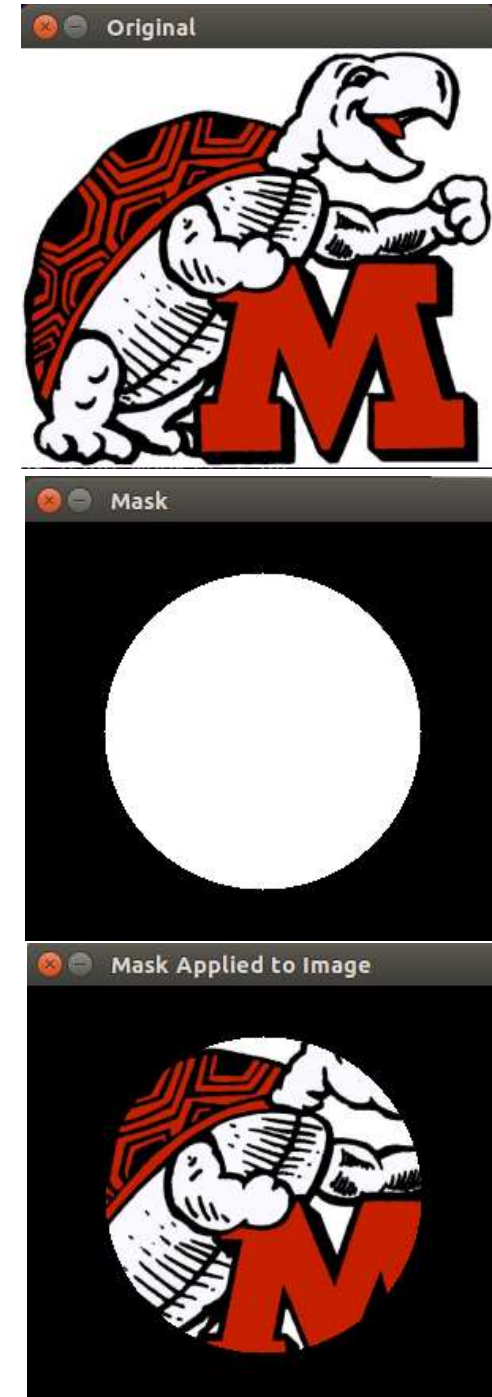
```
import numpy as np
import cv2
import imutils

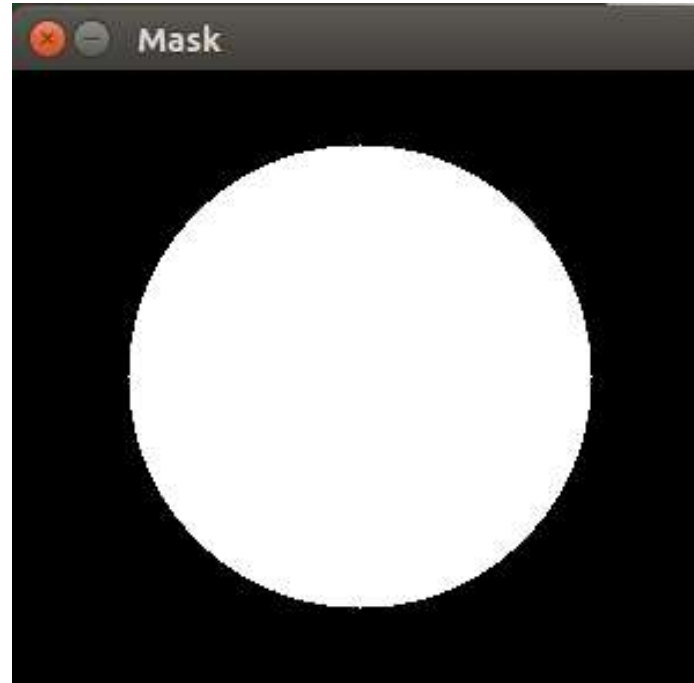
image = cv2.imread("testudo.jpg")
image = imutils.resize(image, width=400)
cv2.imshow("Original", image)

mask = np.zeros(image.shape[:2], dtype = "uint8")
(cX, cY) = (image.shape[1]/2, image.shape[0]/2)
cv2.circle(mask, (cX, cY), 100, 255, -1)
cv2.imshow("Mask", mask)

masked = cv2.bitwise_and(image, image, mask=mask)
cv2.imshow("Mask Applied to Image", masked)

cv2.waitKey(0)
```





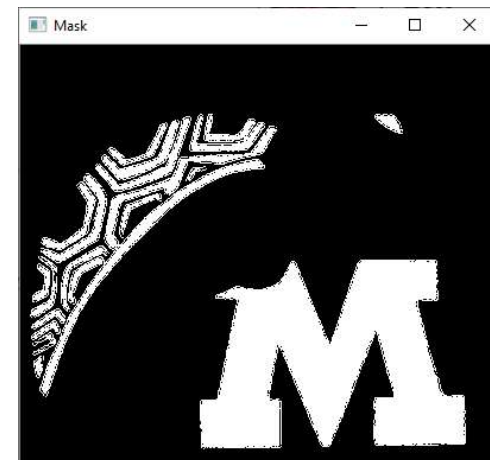
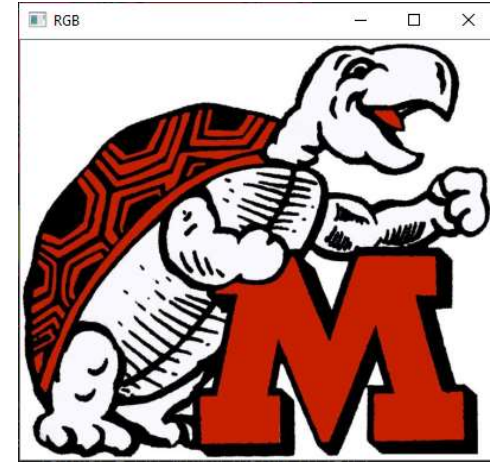
OpenCV Fundamentals

- Color mask

```
import numpy as np
import cv2
import imutils
```

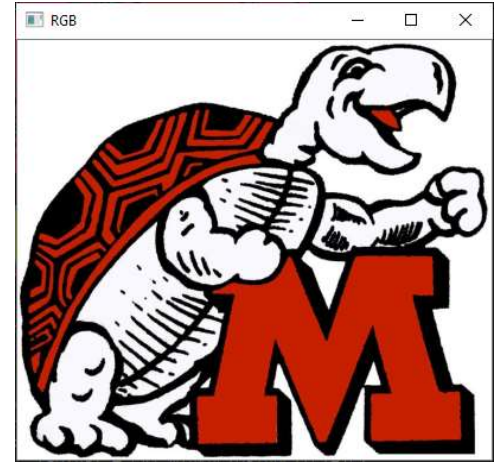
```
image = cv2.imread("testudo.jpg")
image = imutils.resize(image, width=400)
cv2.imshow("BGR", image)
```

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



In-Class Exercise

- Write a Python script that masks for the color “red” inside the Maryland M



OpenCV Fundamentals

- Color mask

```
import numpy as np
import cv2
import imutils

image = cv2.imread("testudo.jpg")
image = imutils.resize(image, width=400)
cv2.imshow("BGR", image)

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
cv2.imshow("HSV", hsv)

cv2.waitKey(0)
```



References

- *Practical Python and OpenCV*, Rosebrock 2016
- *OpenCV Tutorials*
 - <http://docs.opencv.org/2.4/doc/tutorials/tutorials.html>