Project Presentation

Submit 4 pages of report

- Introduction
- Literature review (please show state of the art research work, it is ok if your work is not that good)
- Methodology
- Experimental Results
- Conclusion

OpenCV Tutorial

Dr. Mongkol Ekpanyapong

Image Processing

But the camera sees this:

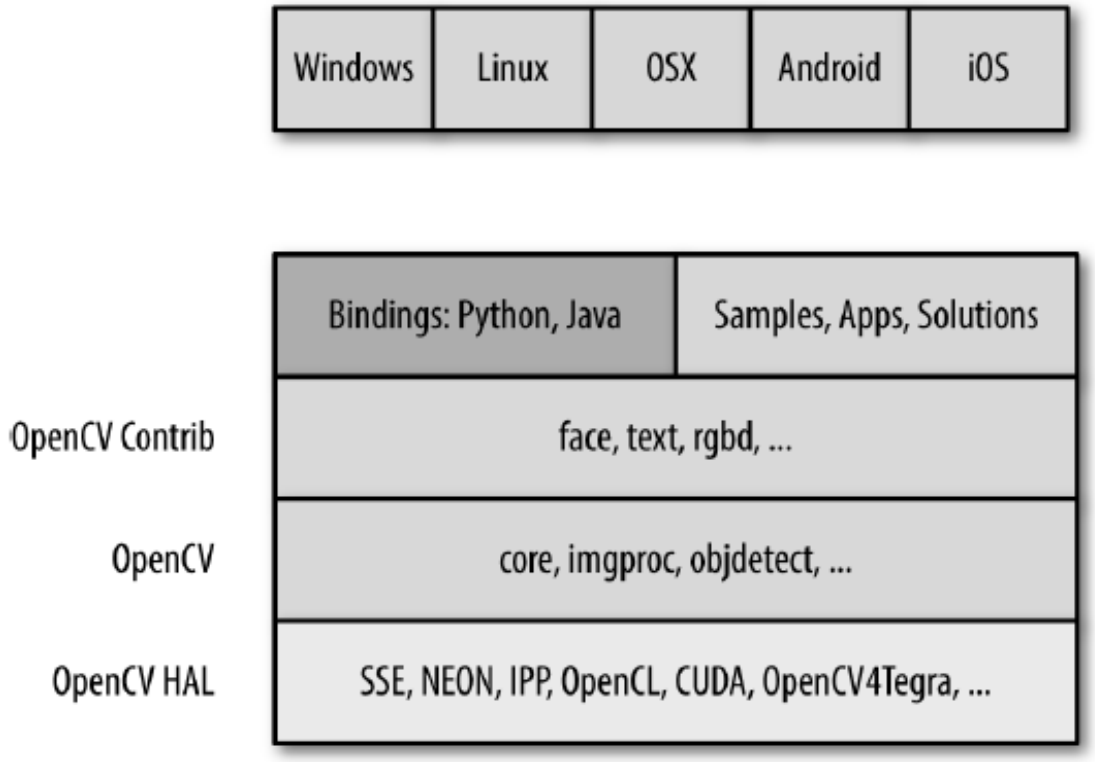
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	74	65
20	41	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

To a computer, the car's side mirror is just a grid of numbers

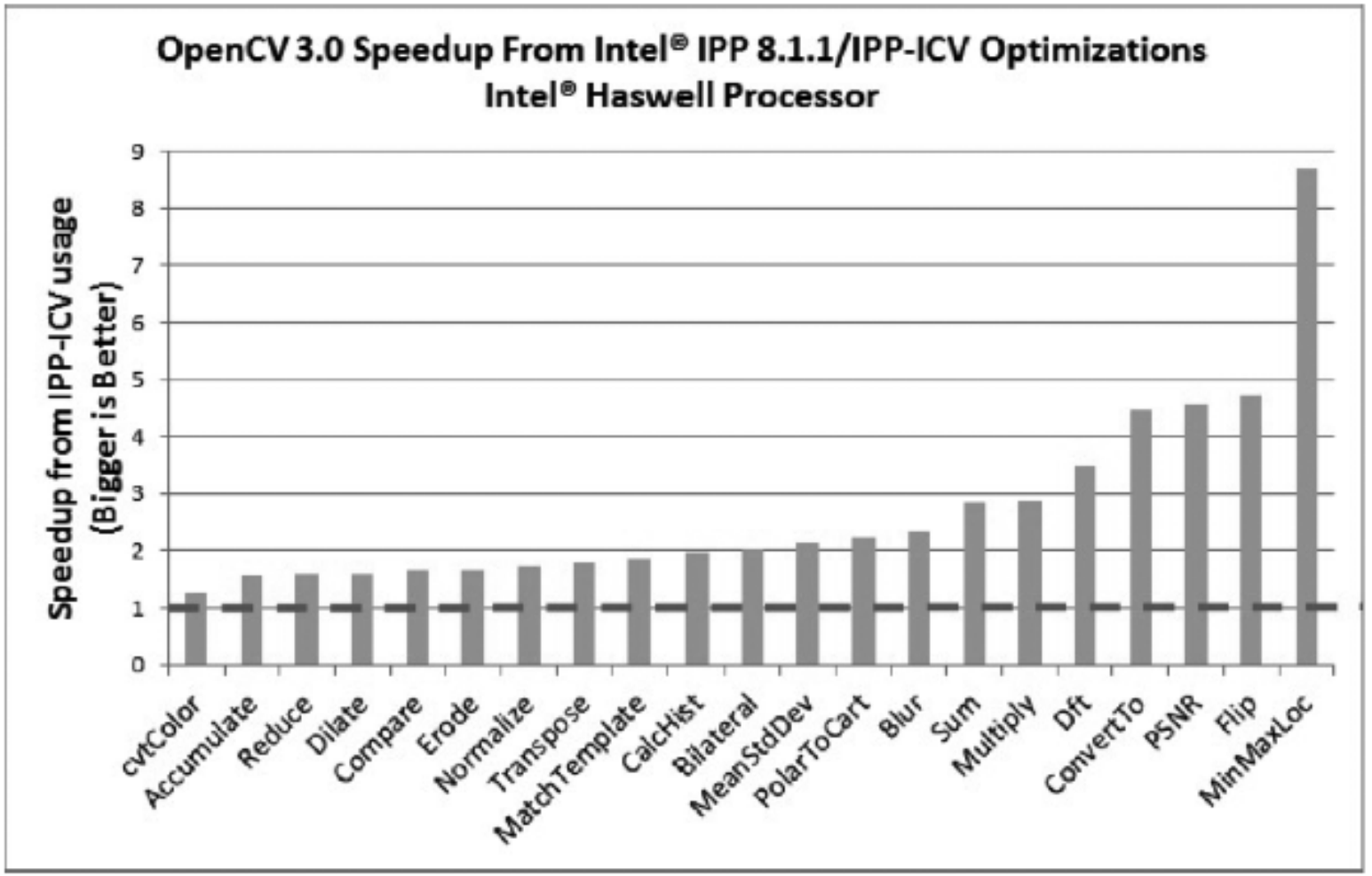
OpenCV

- The project started in 1999 and the first official release was in 2006
- Starting from Intel Research Laboratory as a way to make computer vision infrastructure universally available
- Later, Willow Garage, a robotic research institute was now actively supporting OpenCV (Moved outside Intel)
- In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org,

OpenCV Block Diagram



Performance Speedup using IPPICV library



OpenCV Library

- core is the section contains basic object types
- imgproc is the basic image processing module
- highgui (imgcodes, videoio, and highgui) is for user interface, image codes
- video is for reading video streams
- calib3d is the module to calibrate cameras
- feature2d contains algorithm for features detection
- ml is for machine learning library
- gpu is GPU library
- nonfree are patented algorithms

Portability

	x86/x64	ARM	Other: MIPs, PPC
Windows	SIMD, IPP, Parallel, I/O	SIMD, Parallel (3.0), I/O	N/A
Linux	SIMD, IPP, Parallel,[a] I/O	SIMD, Parallel,[a] I/O	Parallel,[a] I/O*
Android	SIMD, IPP (3.0), Parallel,[b] I/O	SIMD, Parallel,[b] I/O	MIPS—basic support
OS X/iOS	SIMD, IPP (3.0), Parallel, I/O	SIMD, Parallel, I/O	N/A
Other: BSD, QNX, ...	SIMD	SIMD	

[a] Parallelization in Linux is done via a third-party library or by enabling OpenMP.

[b] Parallelization in Android is done via Intel TBB.

Simple OpenCV Program

```
import cv2

image = cv2.imread(“images/size13x13.png")
cv2.imshow(“image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Read image size13x13.png

An 8-bit Representation of an Image

An 8-bit representation of an image in RGB color and grayscale

Program with Printing Value

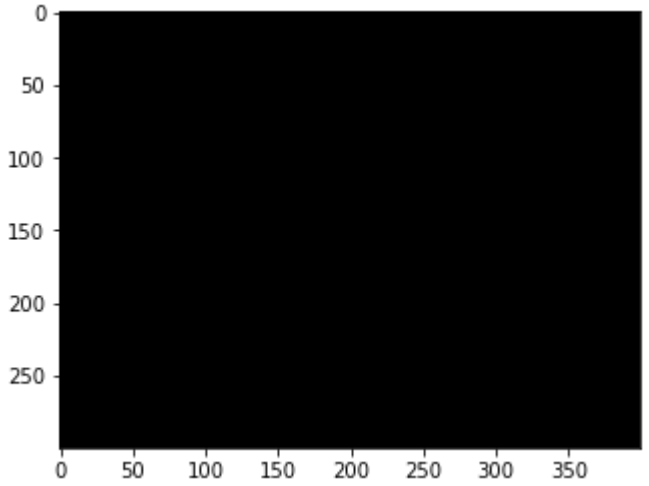
```
import cv2

image = cv2.imread("images/size13x13.png")
cv2.imshow("image", image)

for i in range(0,13):
    for j in range(0,13):
        print("element x ", i, " y ",j," val ", image[i,j,1])

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

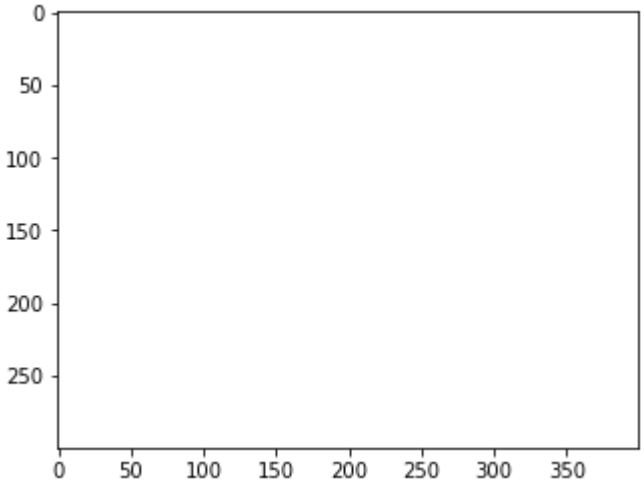
Challenge

- Try to make image: size13x13.png blue color?

Black White Image

```
import cv2
import numpy
import os
# height, width
blackImage =  numpy.zeros((300,400))
cv2.imshow("image black", blackImage)

whiteImage =  255*numpy.ones((300,400))
cv2.imshow("image white", whiteImage)
cv2.waitKey(0)
cv2.destroyAllWindows()
```





Create Random Image

```
import cv2
import numpy
import os

randomByteArray = bytearray(os.urandom(120000))
flatNumpyArray =  numpy.array(randomByteArray)
grayImage = flatNumpyArray.reshape(300,400)
cv2.imshow("image gray", grayImage)

bgrImage = flatNumpyArray.reshape(100,400,3)
cv2.imshow("image bgr", bgrImage)

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

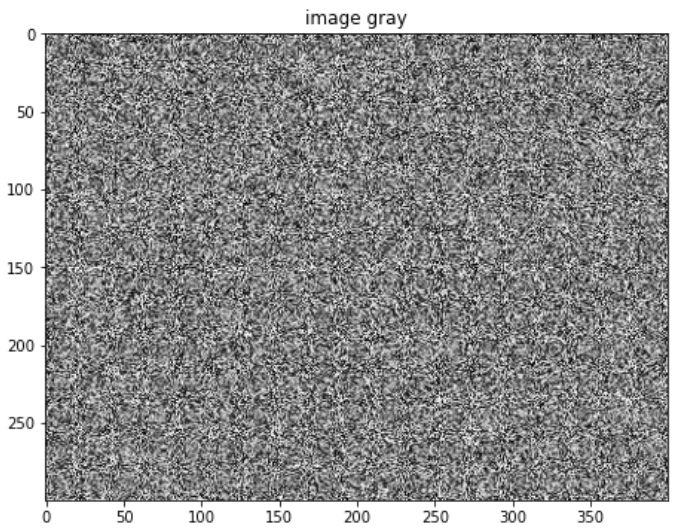


Image Addition

- Addition is used to blend the pixel contents from two images or add a constant value to pixel values of an image
- Adding the contents of two images causes their contents to blend
- Adding a constant value causes an increase or decrease in its overall brightness

Exceeding maximum pixel value

- When adding two images, the values at each pixel can exceed the maximum pixel value. There are two ways to handle this overflow issue:
  - Normalization: Let f is the original intensity value, f_{min} and f_{max} is the minimum and maximum value of the original, L_{max} is the maximum possible intensity value (e.g., 255 for uint 8). The new value g is

    $$g = L_{max} (f - f_{min}) / (f_{max} - f_{min})$$

  - Truncation: simply limiting the results to the maximum positive number

Example

X and Y are unsigned integers 8-bit (uint8)

- $X = \begin{bmatrix} 200 & 100 & 100 \\ 0 & 10 & 50 \\ 50 & 250 & 120 \end{bmatrix}$

- $Y = \begin{bmatrix} 100 & 220 & 230 \\ 45 & 95 & 120 \\ 205 & 100 & 0 \end{bmatrix}$

Example

Calculate : Z = X + Y

First, calculate W (uint 16) = X + Y

$$W = \begin{bmatrix} 300 & 320 & 330 \\ 45 & 105 & 170 \\ 255 & 350 & 120 \end{bmatrix}$$

Example

- A) Normalizing from [45,350] to [0,255]

$$Z_A = \begin{bmatrix} 213 & 230 & 238 \\ 0 & 50 & 105 \\ 175 & 255 & 63 \end{bmatrix}$$

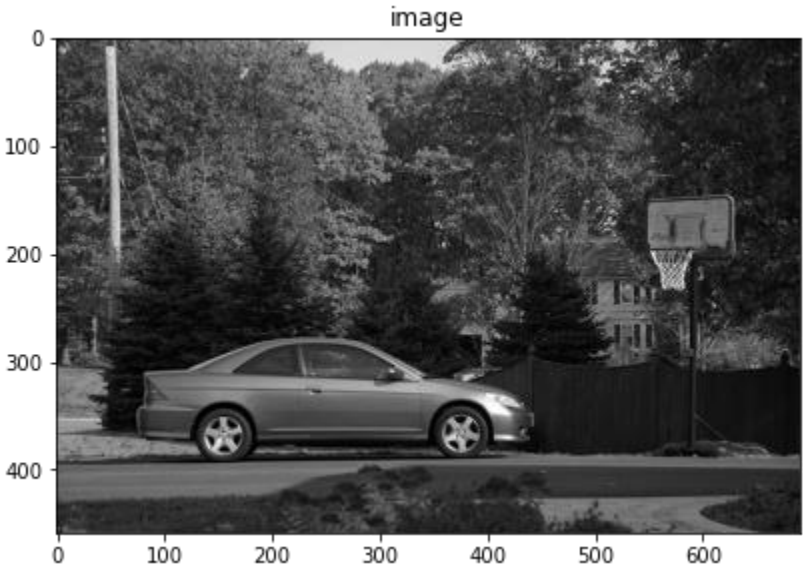
- B) Truncate

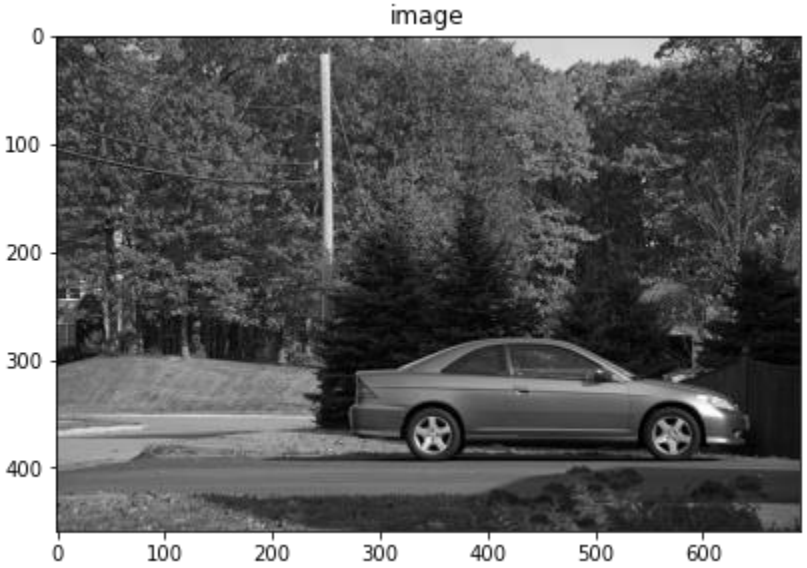
$$Z_B = \begin{bmatrix} 255 & 255 & 255 \\ 45 & 105 & 170 \\ 255 & 255 & 120 \end{bmatrix}$$

In OpenCV, Truncate approach is used.

Add Operation

```
import cv2
import numpy as np

image1 = cv2.imread("images/car-left.tif")
image2 = cv2.imread("images/car-right.tif")

image = cv2.add(image1,image2)
cv2.imshow("image", image)


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





Addition Operation in CV

-OpenCV addition is a saturated operation

-Numpy addition is a modulo operation

Subtraction

- Subtraction is the process opposite to addition
- We can subtract from two images or subtract from a constant
- The default OpenCV is subtract with Saturation

Example

X and Y are unsigned integers 8-bit (uint8)

- $X = \begin{bmatrix} 200 & 100 & 100 \\ 0 & 10 & 50 \\ 50 & 250 & 120 \end{bmatrix}$

- $Y = \begin{bmatrix} 100 & 220 & 230 \\ 45 & 95 & 120 \\ 205 & 100 & 0 \end{bmatrix}$

Example

- A) X-Y

$$Z_A = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 150 & 120 \end{bmatrix}$$

- B) Y- X

$$Z_B = \begin{bmatrix} 0 & 120 & 130 \\ 45 & 85 & 70 \\ 155 & 0 & 0 \end{bmatrix}$$

Example

- C) absdiff(X-Y)

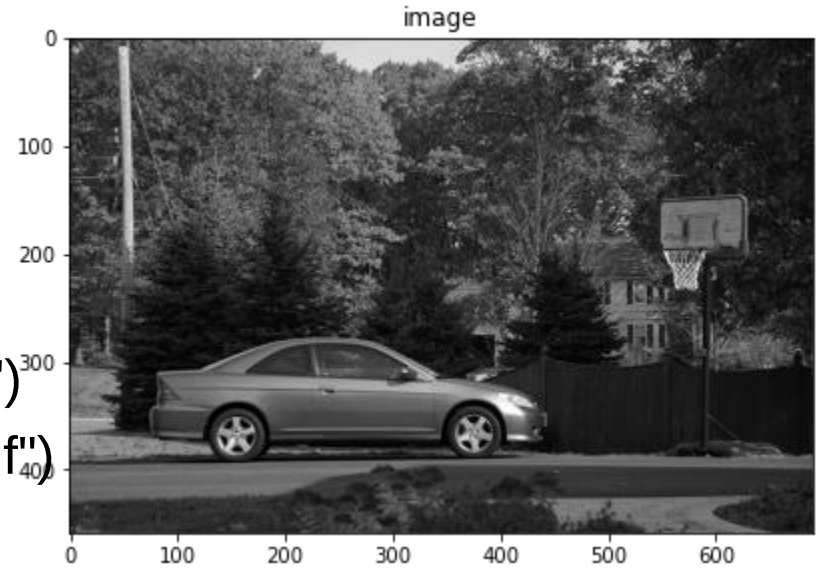
$$Z_C = \begin{bmatrix} 100 & 120 & 130 \\ 45 & 85 & 70 \\ 155 & 150 & 120 \end{bmatrix}$$

Note that you can also do normalization for Z_A and Z_B

In OpenCV, Truncate approach is used.

Subtract Operation

```
import cv2
import numpy as np

image1 = cv2.imread("images/car-left.tif")
image2 = cv2.imread("images/car-right.tif")

image = cv2.subtract(image1,image2)
cv2.imshow("image", image)

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



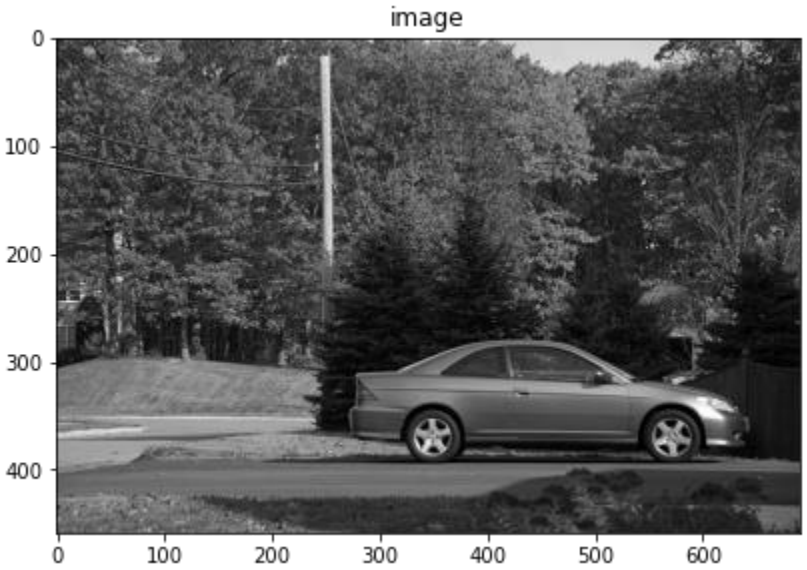


Image Concatenation

```
import cv2
import numpy as np

image1 = cv2.imread("images/car-left.tif")
image2 = cv2.imread("images/car-right.tif")

vis = np.concatenate((image1, image2), axis=1)

cv2.imshow("image", vis)

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

Logic Operations

- Perform bitwise operations on the binary contents of each pixel value
  - AND, OR, XOR require two arguments
  - NOT operator only requires one argument

Logic operations

X

Y

NOT X

NOT Y

X AND Y

X OR Y

X XOR Y

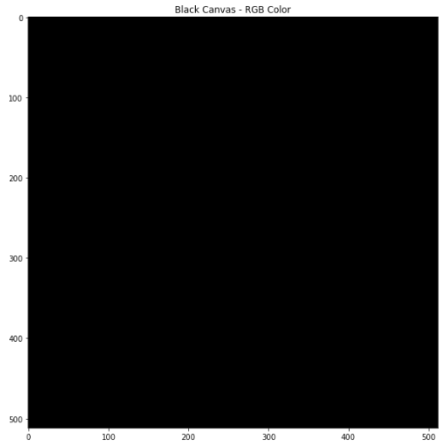
(NOT X) AND Y

Bitwise Operation

```
import cv2
import numpy as np

image1 = cv2.imread("images/wirebond-mask.tif")
height,width,depth = image1.shape
img2 = np.zeros((height,width), np.uint8)
circle_img = cv2.circle(img2,(int(width/2),int(height/2)),200,255,-1)
cv2.imshow("original", image1)

img1_fg = cv2.bitwise_and(image1,image1,mask = circle_img)

cv2.imshow("image", img1_fg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

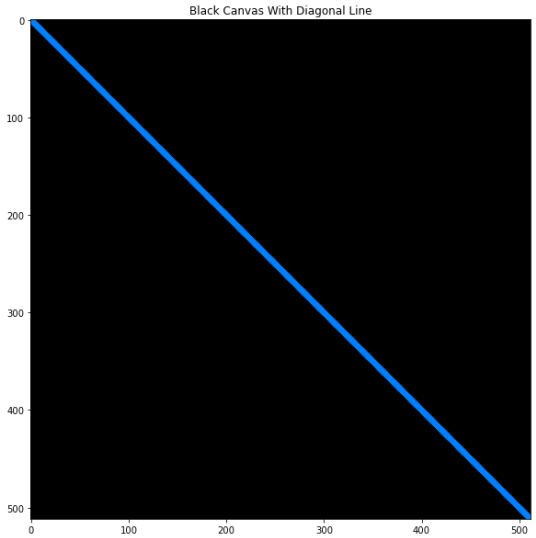
Drawing Canvas

```
# Our Setup, Import Libraries, Create our imshow Function and Download our Images
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Define our imshow function
def imshow(title = "Image", image = None, size = 10):
    w, h = image.shape[0], image.shape[1]
    aspect_ratio = w/h
    plt.figure(figsize=(size * aspect_ratio,size))
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title(title)

image = np.zeros((512,512,3), np.uint8)
imshow("Black Canvas - RGB Color", image)
```



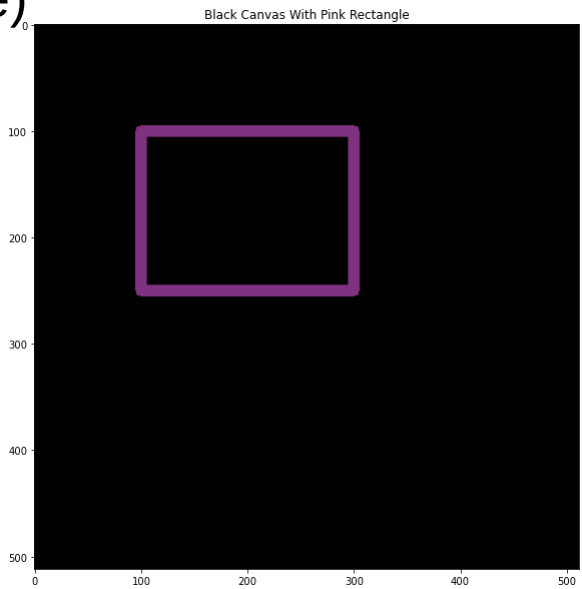
Drawing Line

```
# cv2.line(image, starting cordinates, ending cordinates, color, thickness)


# Note this is an inplace operation, meaning it changes the input image
# Unlike many other OpenCV functions that return a new image leaving the input unaffected
# Remember our image was the black canvas
cv2.line(image, (0,0), (511,511), (255,127,0), 5)

imshow("Black Canvas With Diagonal Line", image)
```



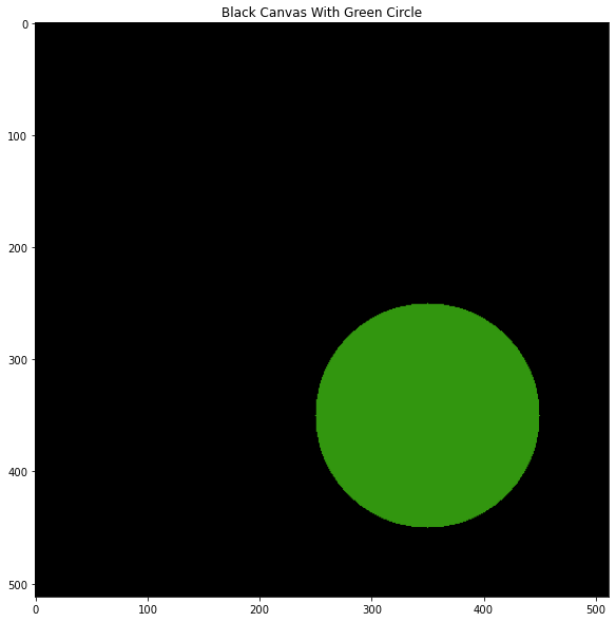
Drawing Rectangle

```
# cv2.rectangle(image, starting vertex, opposite vertex, color, thickness)
# Create our black canvas again because now it has a line in it
image = np.zeros((512,512,3), np.uint8)
# Thickness - if positive. Negative thickness means that it is filled
cv2.rectangle(image, (100,100), (300,250), (127,50,127), 10)
imshow("Black Canvas With Pink Rectangle", image)
```



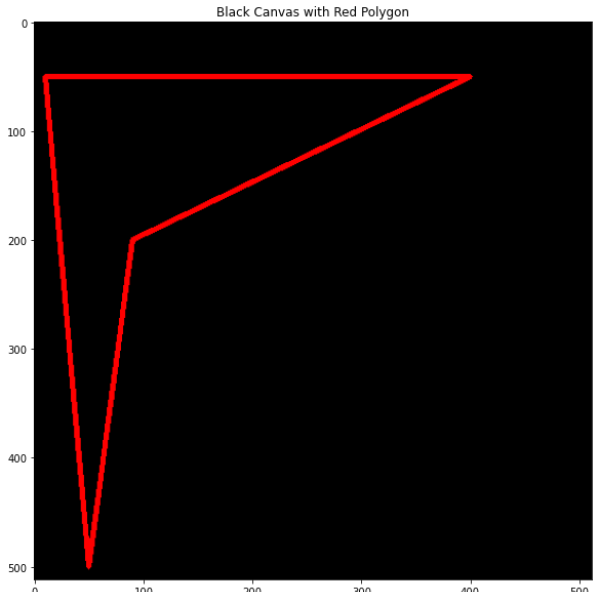
Drawing Circle

```
#  cv2.circle(image, center, radius, color, fill)

image = np.zeros((512,512,3), np.uint8)

cv2.circle(image, (350, 350), 100, (15,150,50), -1)
imshow("Black Canvas With Green Circle", image)
```



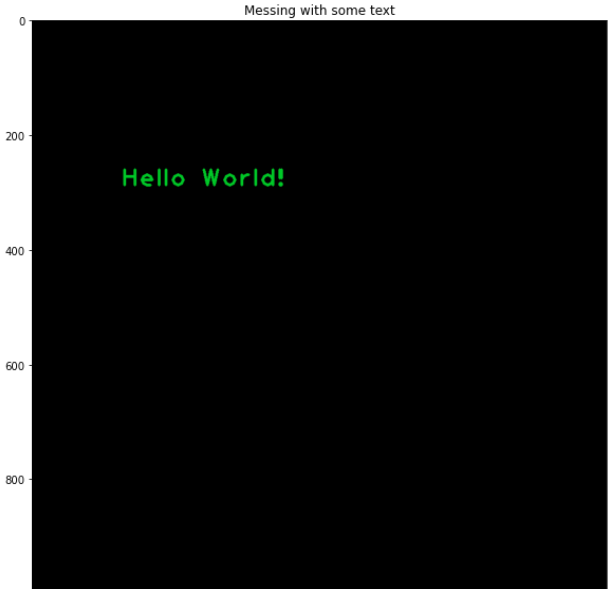
Drawing Polygon

```
# cv2.polylines(image, points, Closed?, color, thickness)
# if Closed = True, we join the first and last points.


image = np.zeros((512,512,3), np.uint8)

# Let's define four points
pts = np.array( [[10,50], [400,50], [90,200], [50,500]], np.int32)

# Let's now reshape our points in form  required by polylines
pts = pts.reshape((-1,1,2))

cv2.polylines(image, [pts], True, (0,0,255), 3)
imshow("Black Canvas with Red Polygon", image)
```



Drawing Text

```
#  cv2.putText(image, 'Text to Display', bottom left starting point, Font, Font Size, Color, Thickness)
# FONT_HERSHEY_SIMPLEX, FONT_HERSHEY_PLAIN
# FONT_HERSHEY_DUPLEX,FONT_HERSHEY_COMPLEX
# FONT_HERSHEY_TRIPLEX, FONT_HERSHEY_COMPLEX_SMALL
# FONT_HERSHEY_SCRIPT_SIMPLEX
# FONT_HERSHEY_SCRIPT_COMPLEX

image = np.zeros((1000,1000,3), np.uint8)
ourString =  'Hello World!'
cv2.putText(image, ourString, (155,290),
cv2.FONT_HERSHEY_PLAIN , 3, (40,200,0), 4)
imshow("Messing with some text", image)
```



User Interaction

```
import cv2
import numpy as np
# mouse callback function
def draw_circle(event,x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        cv2.circle(img,(x,y),100,(255,0,0),-1)

# Create a black image, a window and bind the function to window
img = np.zeros((512,512,3), np.uint8)
cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_circle)
while(1):
    cv2.imshow('image',img)
    if cv2.waitKey(20) & 0xFF == 27:
        break
cv2.destroyAllWindows()
```

Reading Video

```
import cv2
import numpy as np
cap = cv2.VideoCapture("rtsp://admin:pass@10.43.64.94/rtsph2641080p")

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Display the resulting frame
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Questions?

Homework

- Save 100 images of vehicles on the campus on this camera and

  draw rectangle on top of each vehicle

rtsp://admin:Test@1234@10.43.64.61/axis-media/media.amp