Computer Vision for Embedded Systems

Yung-Hsiang Lu
Purdue University
yunglu@purdue.edu

PURDUE UNIVERSITY

Short Biography

- Professor, Electrical and Computer Engineering
- Director, Purdue John Martinson Entrepreneurial Center (2020-2022)
- Purdue University Faculty Scholar (2021-2026)
- Research topics: computer systems, computer vision
- IEEE Fellow, ACM Distinguished Scientist and Speaker
- Published 170+ papers. Citations ~ 9,000, h-index 39
- 5 issued patents
- PhD EE Stanford; BSEE National Taiwan University
- https://yhlu.net/

What is this (graduate) course?

What will we do?

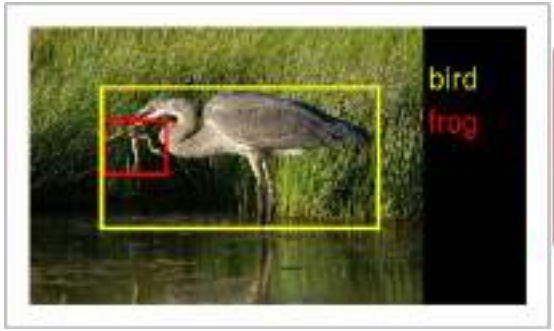
- Develop computer vision programs
- Learn OpenCV and PyTorch implementation
- Evaluate and improve performance (including execution time)
- Read recent papers about efficient computer vision
- Investigate business opportunities of the vision technologies

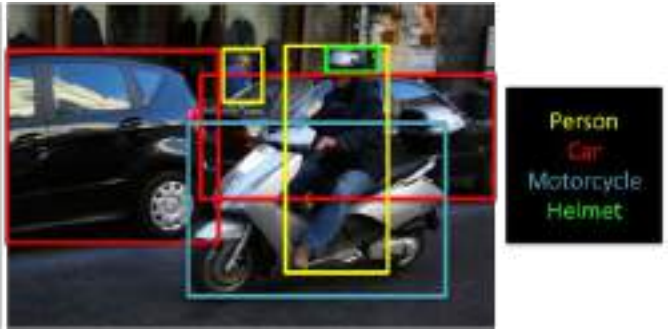
What will we not do?

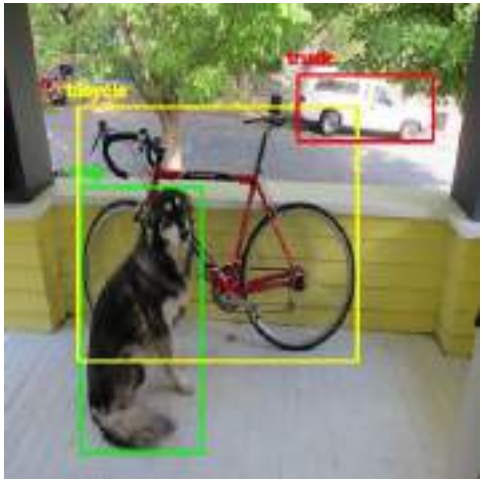
- Derive mathematical equations without implementation
- Explain code line by line
- Use supercomputers; Design autonomous cars

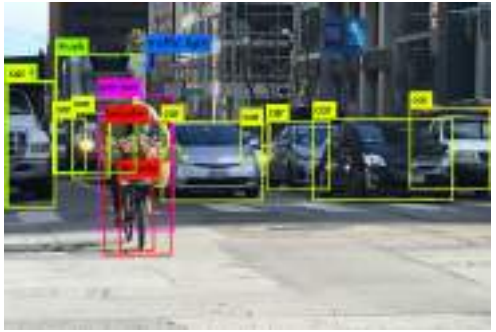
Computer Vision

Use computer programs to understand images or video
(this course focuses on images)









https://image-net.org/
https://github.com/tejaslodaya/car-detection-yolo
https://towardsdatascience.com/yolo-you-only-look-once-17f9280a47b0

Computer Vision may give "meanings"

https://www.crcv.ucf.edu/data/UCF_Sports_Action.php

Computer Vision using GPU

GPU: Graphics Processing Units

Embedded Systems

Computers "embedded" in systems whose primary functions are ***not*** computing.

https://www.rs-online.com/designspark/applications-of-embedded-systems-1

Why computer vision on embedded systems?

Cameras are everywhere

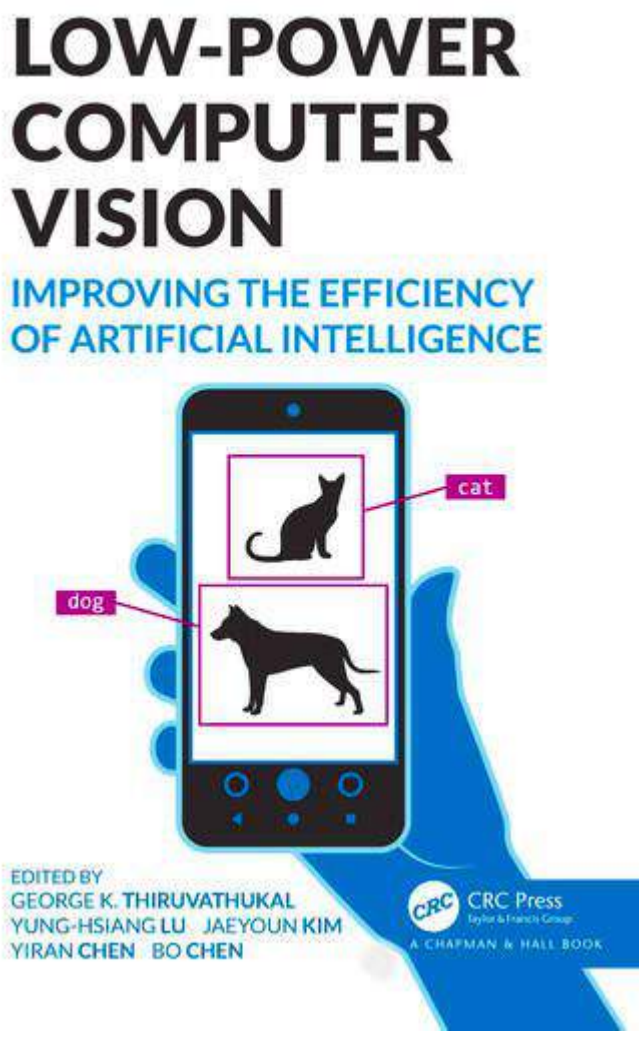
Image Sources

- https://www.nytimes.com/wirecutter/reviews/best-drones/
- https://www.texassurveillance.com/solar-powered-wireless-security-cameras/
- https://www.pcmag.com/news/google-glass-everything-you-need-to-know
- https://www.gim-international.com/content/news/faro-launches-3d-laser-scanning-integration-with-boston-dynamics-mobile-robot
- https://www.foxelli.com/products/foxelli-trail-camera-wildlife-scouting-hunting-camera
- https://www.abcactionnews.com/news/region-pinellas/can-you-tell-the-difference-from-a-red-light-camera-and-a-traffic-monitoring-camera
- https://picavi.com/en/wearables/
- https://www.reddit.com/r/Wevolver/comments/lbtj36/automate_your_counting_for_inventory_shipping_and/

Administration

- One-credit graduate course: three lectures per week for 5 weeks
- Expected background: Python Programming, basic machine learning
- Grading: 85:A, 75:B, 65:C, 50: D. below 50: F
  - 30: 3 homework assignments (individual), 10 each
  - 20: Online discussion (individual): 1 per post
  - 30: 5 online quizzes (individual), 6 each
  - 20: design project (2-people team) : 10 proposal, 10 final presentation

- Repository for sample code: https://github.com/luyunghsiang/embeddedvision
- primary communication channel: piazza.com/purdue/fall2022/ece595
- use email for personal reasons ONLY
- ***If there is an emergency, please dial 911***
- ***NEVER send any message with EMERGENCY (or URGENT) as the subject***



www.routledge.com/9780367744700
Edited By George K. Thiruvathukal, Yung-Hsiang Lu, Jaeyoun Kim, Yiran Chen, Bo Chen
ISBN 9780367744700
438 Pages, 101 Illustrations

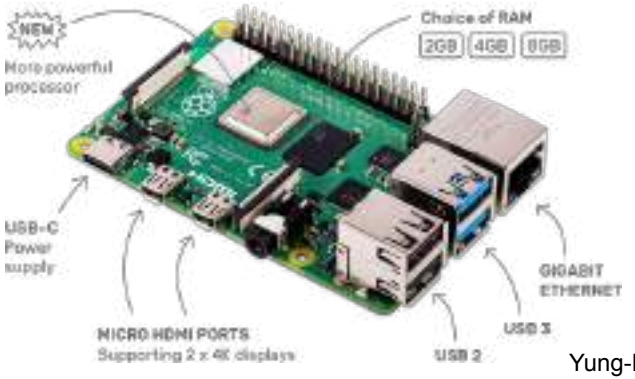
20% Discount code FLA22

CRC Press
Taylor & Francis Group

Optional Hardware

This course uses free Google Colab. Hardware not needed.
If you want to use hardware, you can buy for about $100

1. Raspberry Pi 4 ($35 for 2GB, $75 for 8GB)
2. MicroSD ($10 for 32GB, $25 for 128GB)
3. Power adapter $8
4. Antistatic wrist strap $10
5. Micro HDMI converter $8
6. Raspberry Pi 400 Kit $100



Topics

Lecture	Lecture
01	Overview, data formats, OpenCV, Quantization (assignment)
02	OpenCV Edge detection and segmentation, PyTorch
03	Applications, business opportunities, project
04	Machine learning and neural networks
05	Modular neural networks (assignment)
06	How to review papers (assignment)
07	Quantization in PyTorch, performance and resources

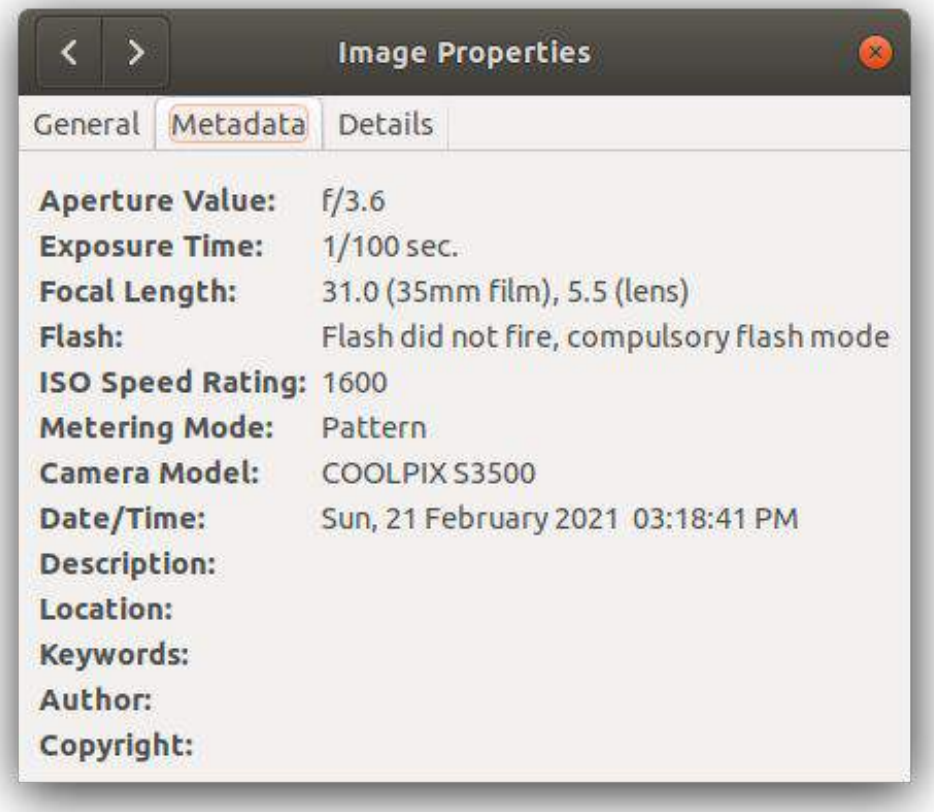
Topics

Week	Lecture
08	Object detection, tracking
09	Data bias and privacy
10	Privacy and crowdsourcing
11	Synthesize data and neural architecture search
12	Transformer-1
13	Transformer-2
14	Real-time scheduling
15	Research topics (Generative Models, Consistency)

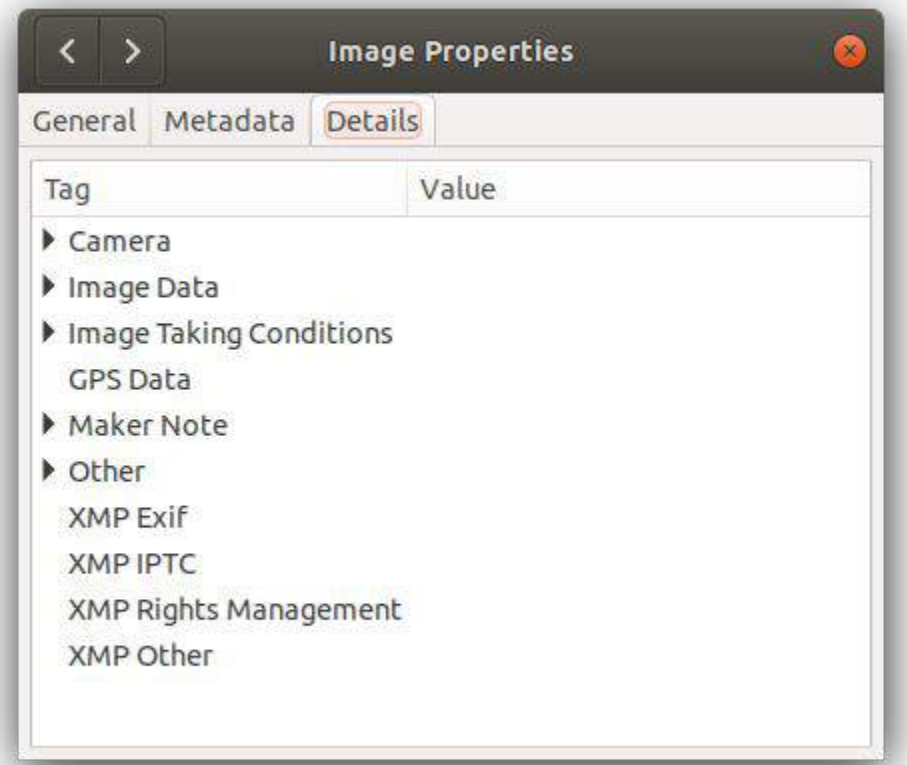
Data Formats / Colors

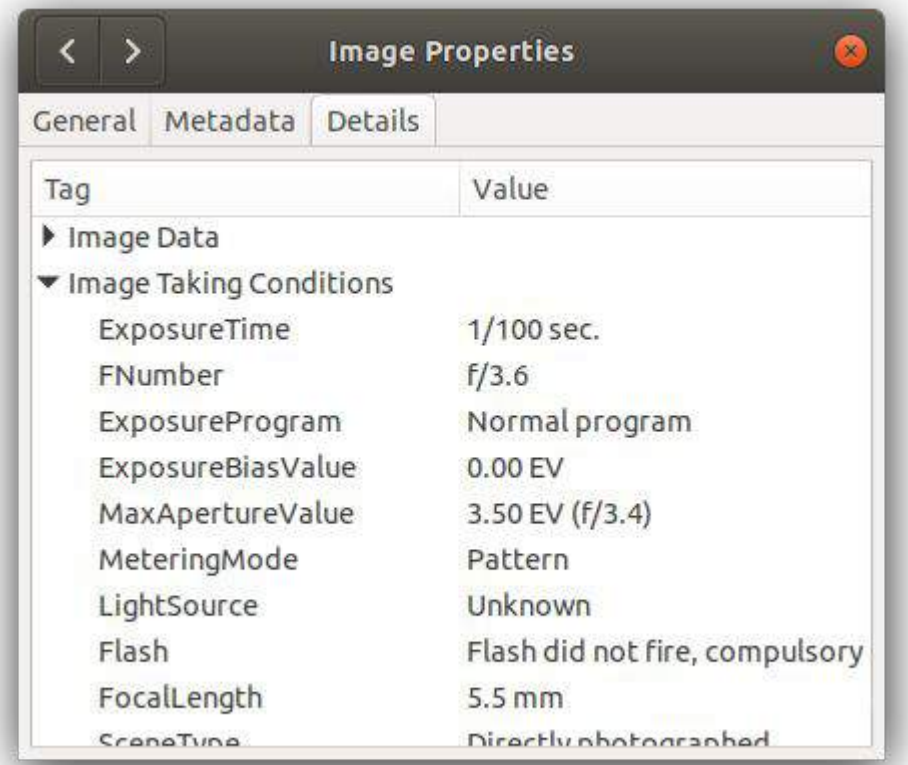
Data Formats: Image

Metadata: description of the data
data (pixels)

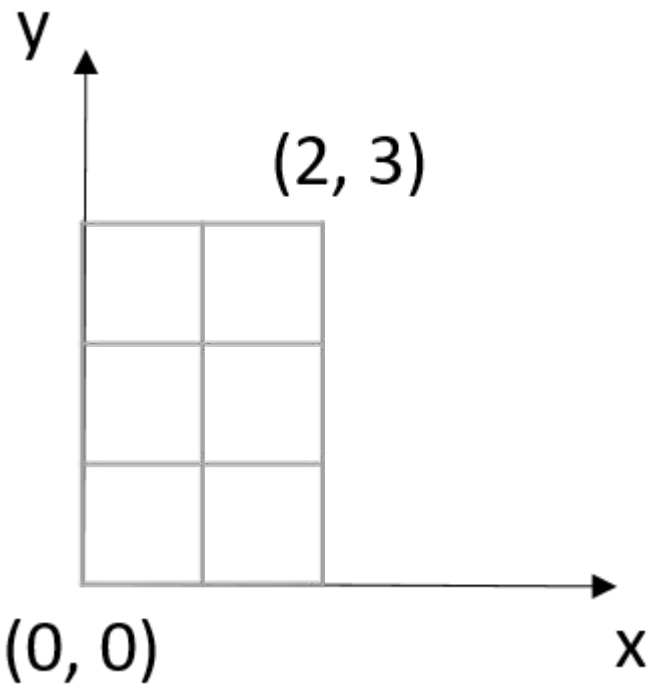


More metadata

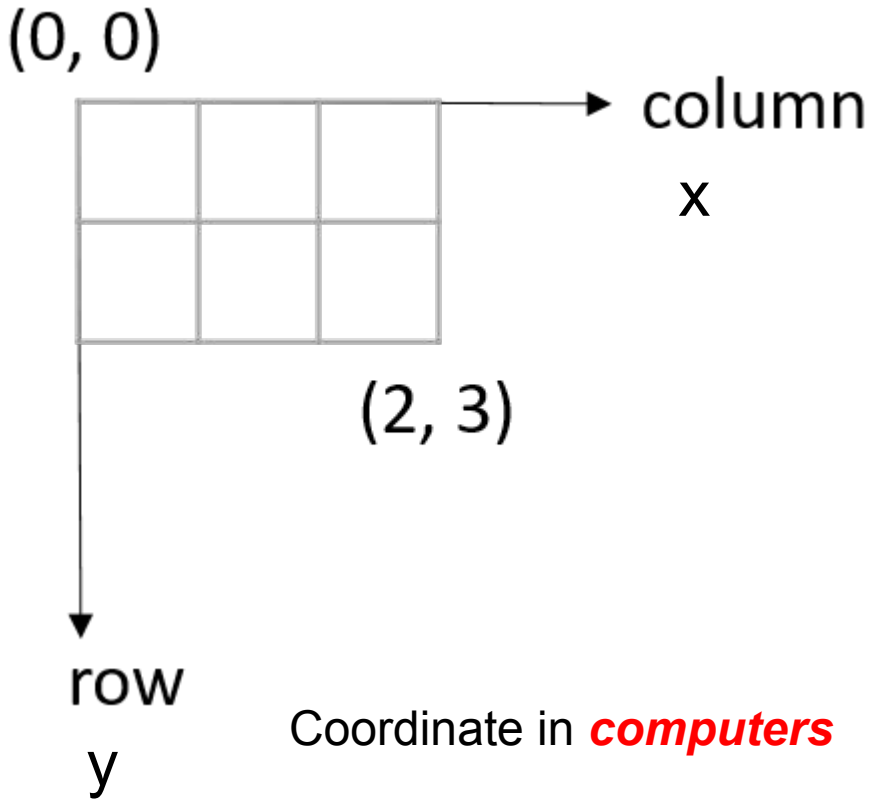




Coordinate Systems

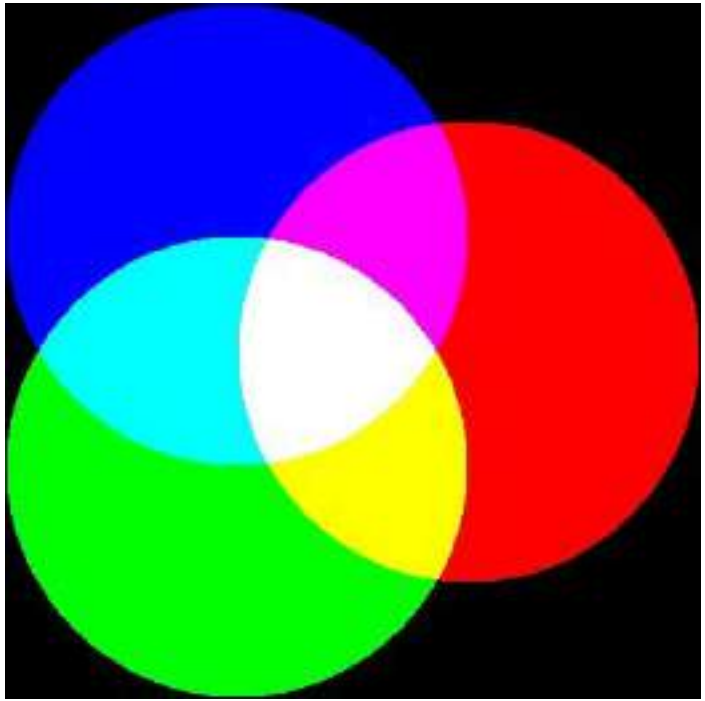


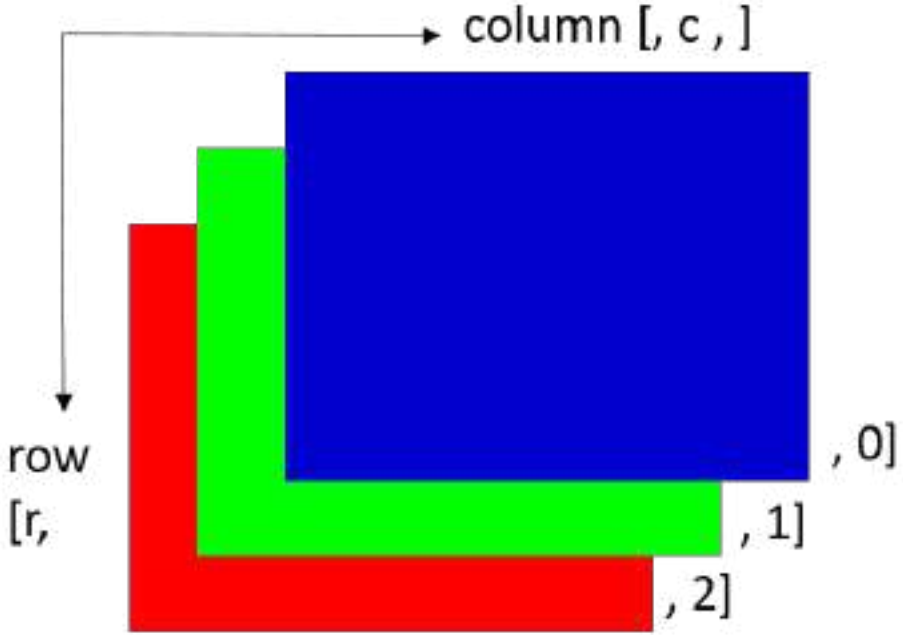
First quadrant in analytical geometry



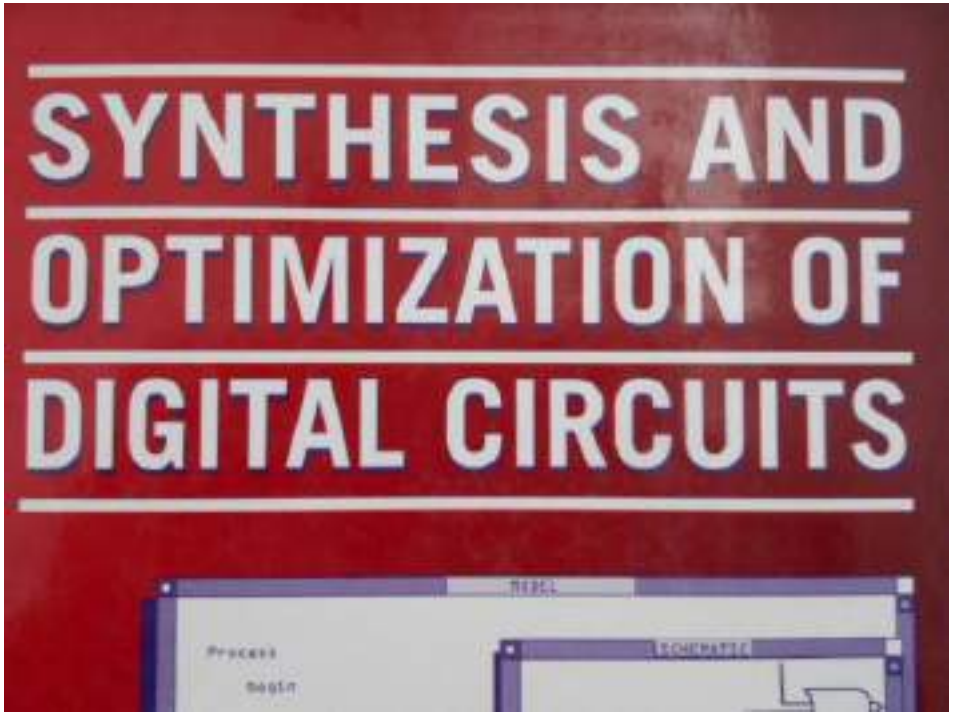
Coordinate in ***computers***

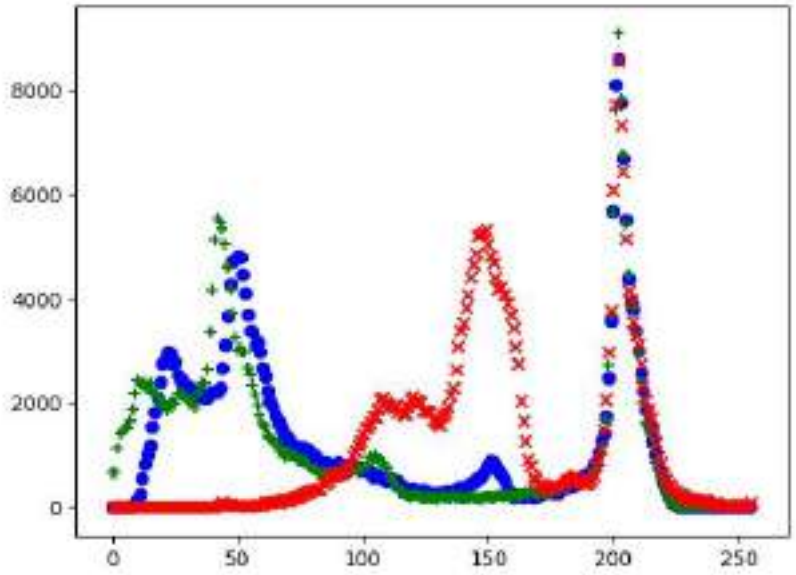
Color Space

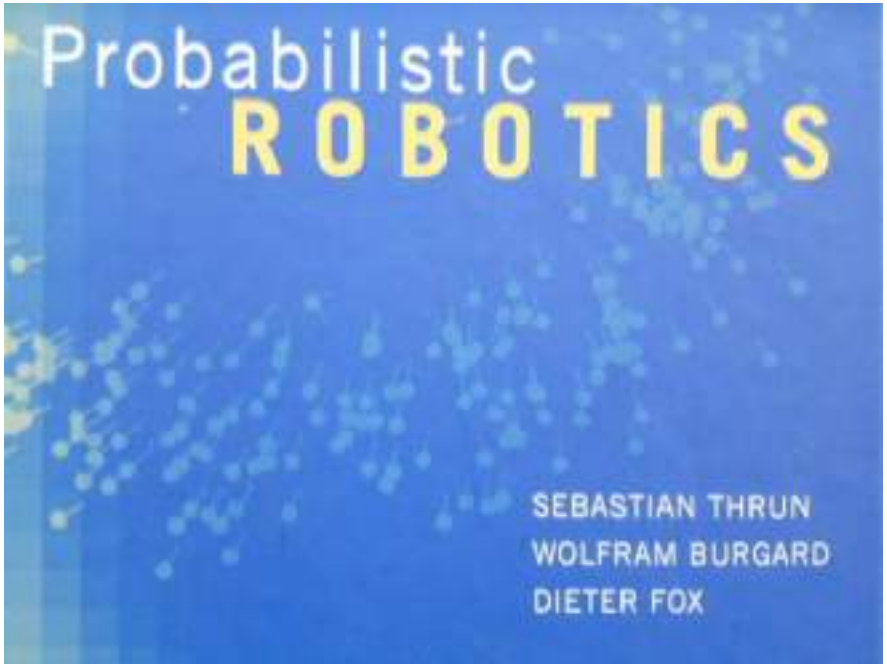


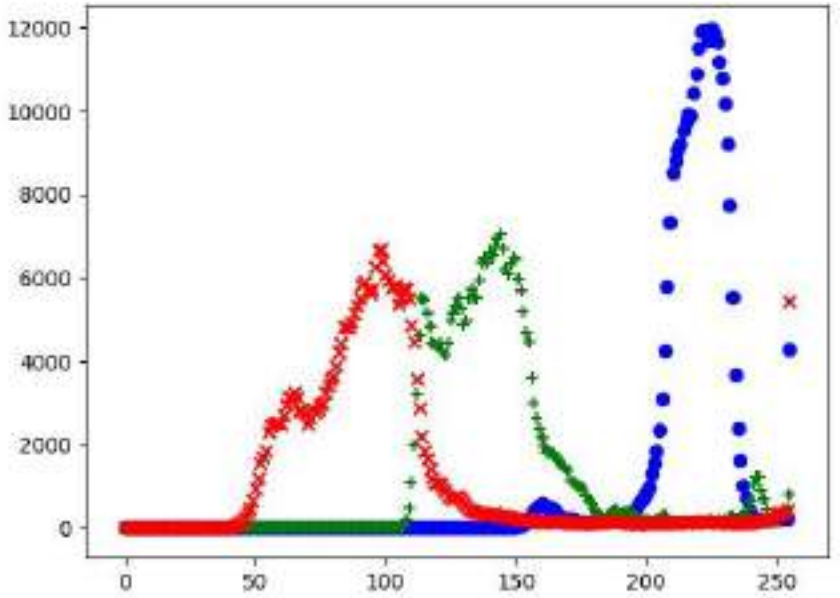


Color Histogram



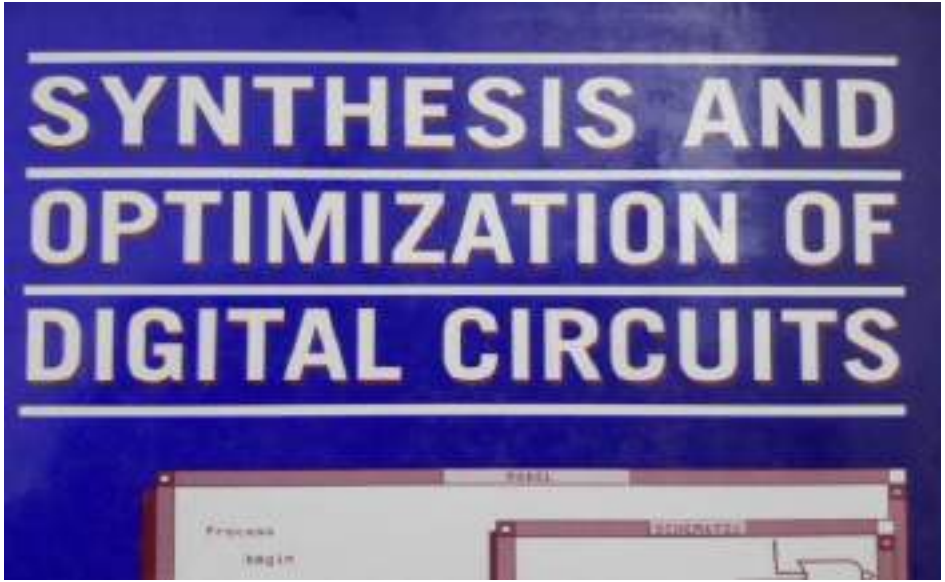


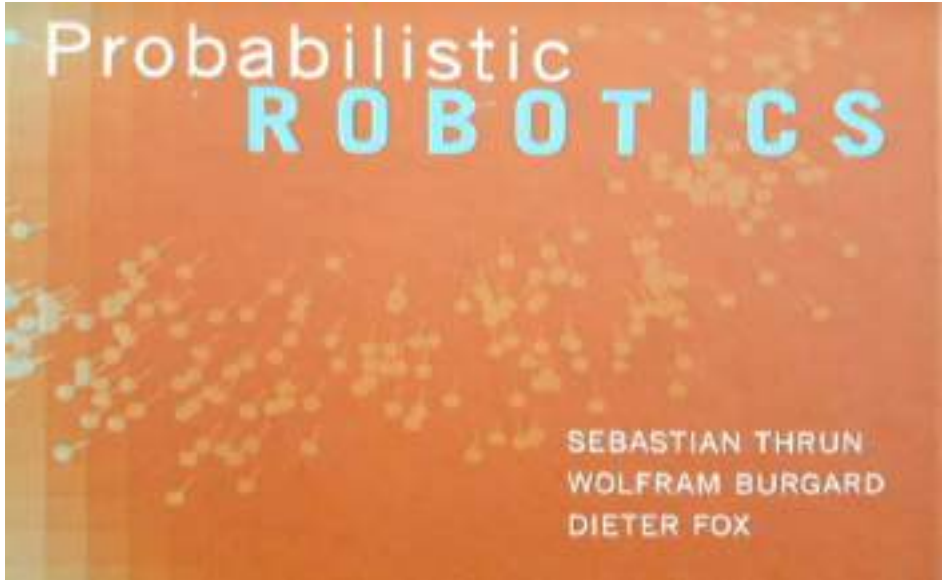

Probabilistic
ROBOTICS
SEBASTIAN THRUN
WOLFRAM BURGARD
DIETER FOX


12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250

Red-Blue Swap

swap red and blue of each pixel

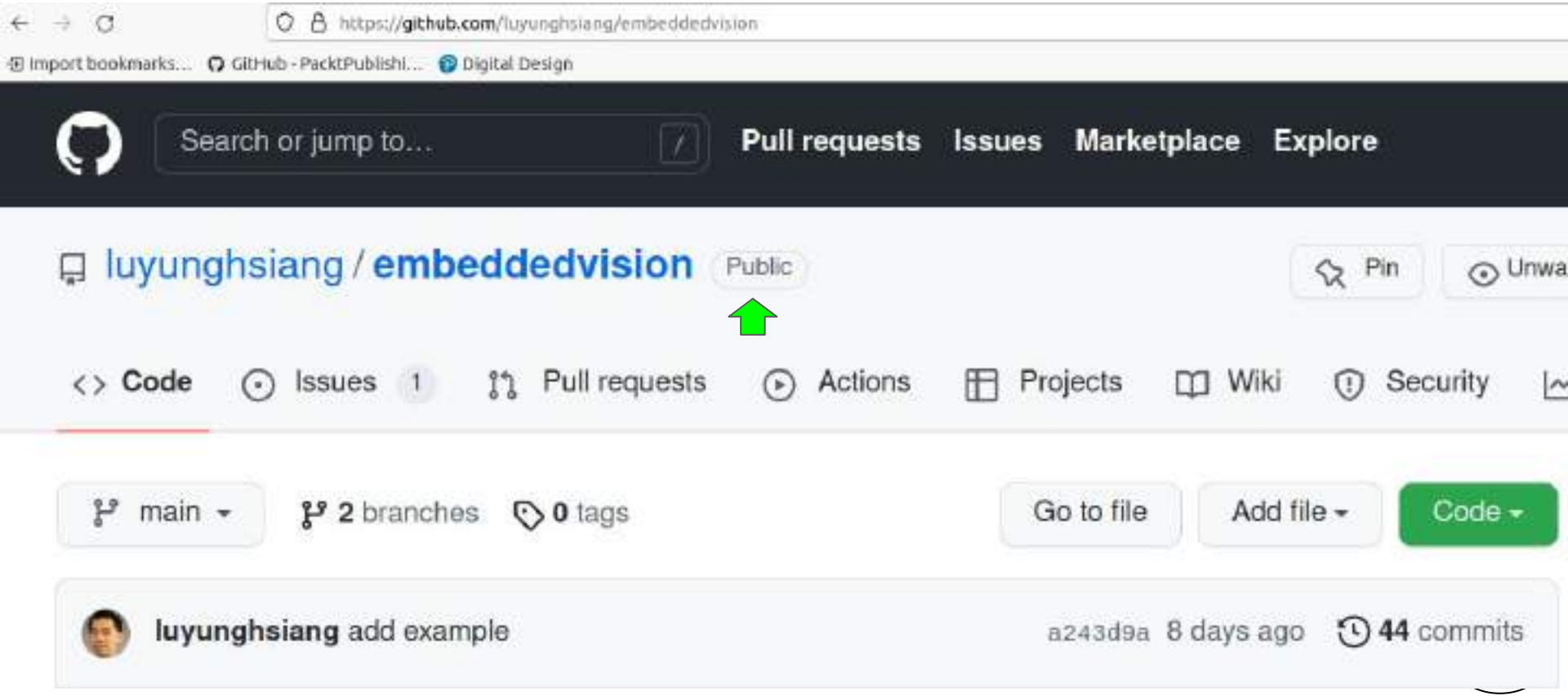




OpenCV (popular before neural networks became widely used)

can be used for pre or post processing of data

git clone git@github.com:luyunghsiang/embeddedvision.git



Read and Show Image

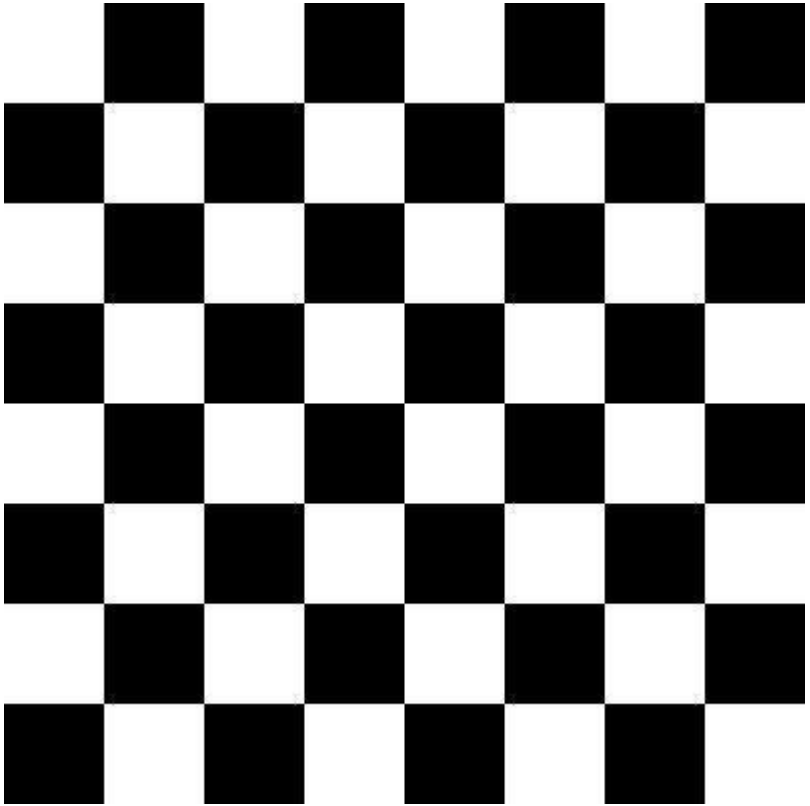
```
# read an image and show it
import cv2
import sys
def showImage(filename):
    img = cv2.imread(filename)
    cv2.imshow("imshow", img)
    k = cv2.waitKey(0)
if __name__ == "__main__":
    if (len(sys.argv) != 2):
        sys.exit('need the name of an image file')
    showImage(sys.argv[1])
```

+

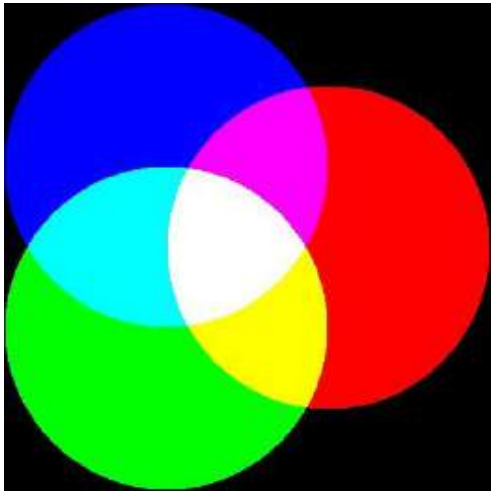
||

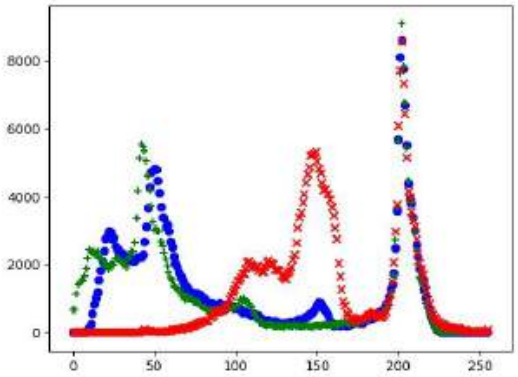
```
# read two images and show the mixed image
import cv2
import sys
def mixImage(filename1, filename2):
    img1 = cv2.imread(filename1)
    img2 = cv2.imread(filename2)
    # only images of the same size can be added
    size1 = img1.shape
    size2 = img2.shape
    height = min(size1[0], size2[0])
    width  = min(size1[1], size2[1])
    print (size1)
    print (size2)
    print ([height, width])
    nimg1 = cv2.resize(img1, [width, height])
    nimg2 = cv2.resize(img2, [width, height])
    img3 = cv2.addWeighted(nimg1, 0.5, nimg2, 0.5, 0)
    cv2.imshow('imshow', img3)
    cv2.imwrite('mixed.jpg', img3)
    k = cv2.waitKey(0)
if __name__ == "__main__":
    if (len(sys.argv) != 3):
        sys.exit('need the name of two image files')
    mixImage(sys.argv[1], sys.argv[2])
```

```
# checkerboard.py
# create a checkerboard

import numpy, cv2
width = 400
eighth = int(width/8)

# create a 3-dimensional array
# first two dimensions for horizontal and vertical
# third dimension for colors (R, G, B)
# zeros make the pixels black
board = numpy.zeros([width, width, 3])

# create an 8 x 8 checkerboard

whitesquare = numpy.ones([eighth, eighth]) * 255

for row in range(8):
    for col in range(8):
        if ((row % 2) == (col % 2)):
            srow = row  * eighth    # start
            erow = srow + eighth    # end
            scol = col  * eighth
            ecol = scol + eighth
            board[srow:erow, scol:ecol, 0] = whitesquare
            board[srow:erow, scol:ecol, 1] = whitesquare
            board[srow:erow, scol:ecol, 2] = whitesquare

cv2.imwrite('checkerboard.jpg', board)
cv2.imshow('checkerboard', board)
cv2.waitKey(0)
```



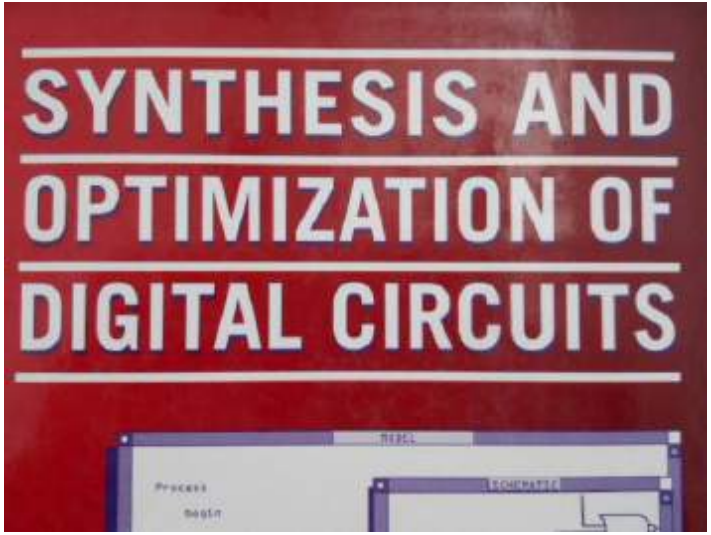
```
# colormix.py
# create three overlapping areas of RGB
import math
import numpy, cv2
width = 400
third = int(width/3)

def drawCircle(crow, ccol, radius, canvas, color):
    # draw a filled cirlce center at (crow, ccol)
    for row in range(radius):
        col = int(math.sqrt(radius * radius - row * row))
        canvas[crow - row: crow + row, ccol - col: ccol + col, color] = 255

canvas = numpy.zeros([width, width, 3])
drawCircle(third, third, third, canvas, 0)
drawCircle(2 * third, third, third, canvas, 1)
drawCircle(int(width / 2), 2 * third, third, canvas, 2)
cv2.imwrite('colormix.jpg', canvas)
```

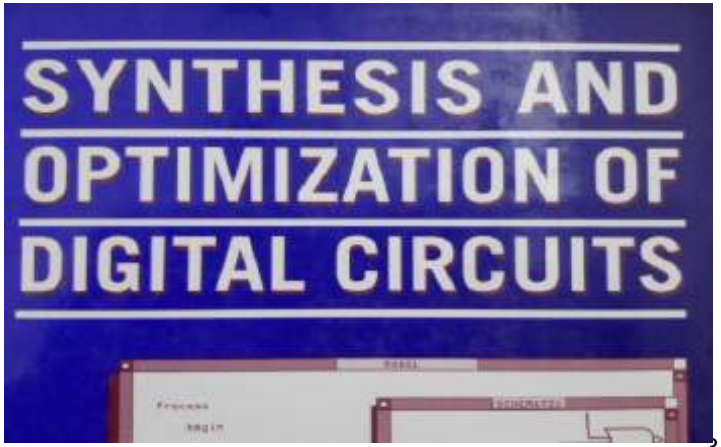


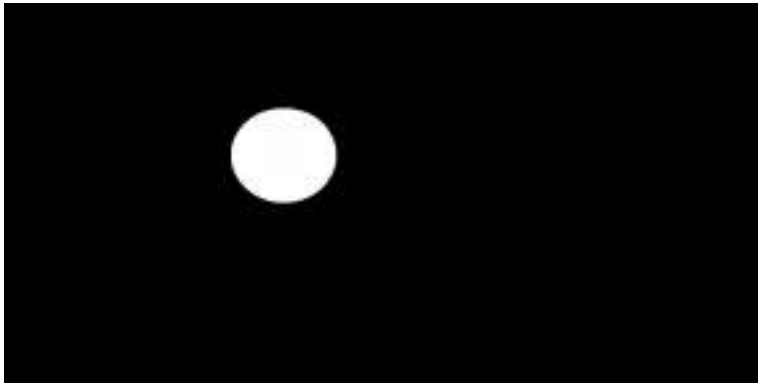


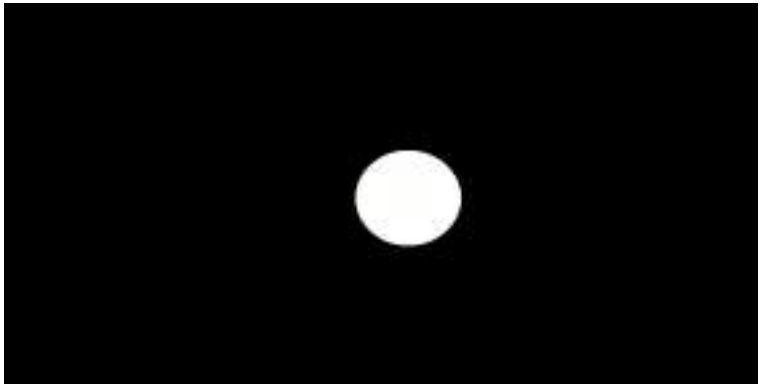

```python
# colorhistogram.py
# draw the histograms or Red, Green, Blue

import numpy, cv2, sys
import matplotlib.pyplot as pyplot

def histogram(image):
    dimension = image.shape
    intensity = numpy.zeros([256, 3])
    for row in range(dimension[0]):
        for col in range(dimension[1]):
            for clr in range(3):
                intensity[image[row, col, clr], clr] += 1
    # plot the three colors
    xaxis = range(256)
    yaxis = range(256)
    # print(len(intensity[:,0]))
    pyplot.scatter(xaxis, intensity[:,0], color = 'b', marker = 'o')
    pyplot.scatter(xaxis, intensity[:,1], color = 'g', marker = '+')
    pyplot.scatter(xaxis, intensity[:,2], color = 'r', marker = 'x')
    pyplot.savefig('colorhistogram.jpg')

if __name__ == "__main__":
    if (len(sys.argv) != 2):
        sys.exit('need the name of an image file')
    image = cv2.imread(sys.argv[1])
    histogram(image)
```

```python
# swapredblue.py
# swap the values of red and blue of each pixel

import numpy, cv2, sys
import matplotlib.pyplot as pyplot

def swapredblue(image):
    dimension = image.shape
    for row in range(dimension[0]):
        for col in range(dimension[1]):
            # read the original values
            blue = image[row, col, 0]
            red  = image[row, col, 2]
            # swap
            image[row, col, 0] = red
            image[row, col, 2] = blue

if __name__ == "__main__":
    if (len(sys.argv) != 2):
        sys.exit('need the name of an image file')
    image = cv2.imread(sys.argv[1])
    swapredblue(image)
    cv2.imwrite('swappedimage.jpg', image)
```





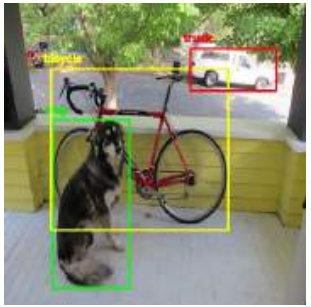




```
# frame2video.py
# create a video from individual frames of images
# a bouncing white ball

import numpy
import math
from cv2 import VideoWriter, VideoWriter_fourcc

width = 1280
height = 720
FPS = 24
seconds = 10
radius = int(height/8)

fourcc = VideoWriter_fourcc(*'MP42')
video = VideoWriter('./ball.avi', fourcc, float(FPS), (width, height))
square = numpy.ones((int(height/4), int(width/4), 3), dtype=numpy.uint8) * 255

# create a white ball
ball = numpy.zeros((2 * radius, 2 * radius, 3), dtype=numpy.uint8)
for row in range(radius):
    col = int(math.sqrt(radius * radius - row * row))
    ball[radius - row: radius + row, radius - col: radius + col, ] = 255
deltarow = int(3 * height / (FPS * seconds))
srow = radius

for count in range(FPS*seconds):
    canvas = numpy.zeros((height, width, 3), dtype = numpy.uint8)
    scol = int (count * 3 * width / (4 * FPS * seconds))
    canvas[srow - radius: srow + radius, scol: scol + 2 * radius,] = ball
    srow += deltarow

    # change direction?
    if ((srow + deltarow) < radius):
        deltarow = - deltarow
    if ((srow + deltarow) >= (height - radius)):
        deltarow = - deltarow

    video.write(canvas)
video.release()
```

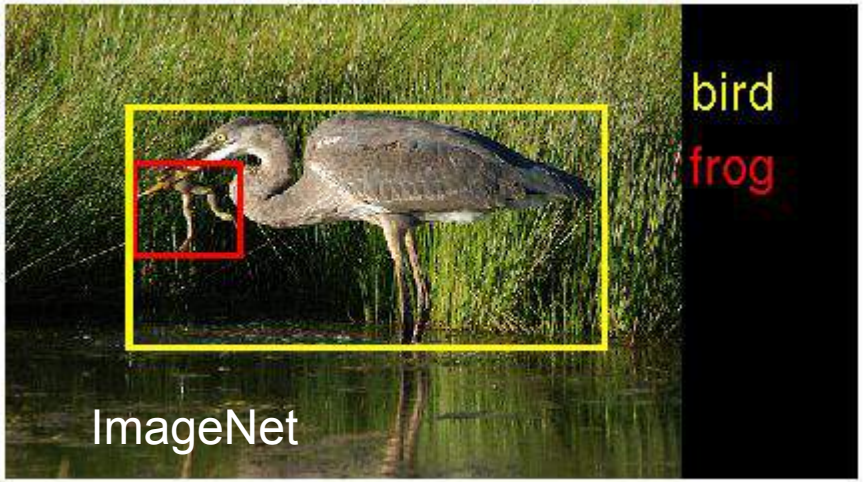
```python
# saveframes.py
# read a video file and save every 100th frames

import numpy, cv2, sys

def saveFrames(video):
    stream = cv2.VideoCapture(video)
    numframe = 0
    while (stream.isOpened()):
        ret, frame = stream.read()
        if (ret == False): # finished all frames
            break
        # read a frame successful
        if ((numframe % 100) == 0):
            filename = 'frame' + str(numframe) + '.jpg'
            cv2.imwrite(filename, frame)
        numframe += 1
    print(numframe)
    stream.release()

if __name__ == "__main__":
    if (len(sys.argv) != 2):
        sys.exit('need the name of a video file')
    saveFrames(sys.argv[1])
```

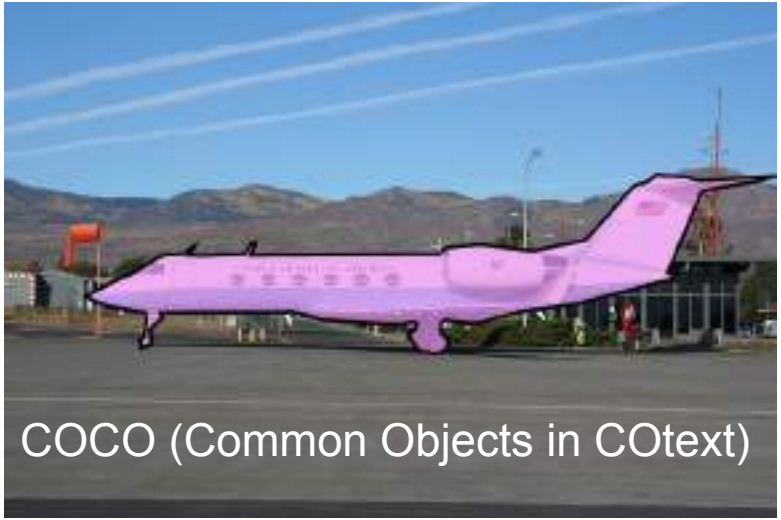
```python
# text.py
# add text to an image

import numpy
import sys
import cv2

def addText(filename):
    rgbimage = cv2.imread(filename)
    font = cv2.FONT_HERSHEY_SIMPLEX
    color = (0, 255, 0) # green
    textimage = cv2.putText(rgbimage, 'OpenCV', (100, 100), font, 1, color)
    cv2.imwrite("text.jpg", textimage)

if __name__ == "__main__":
    if (len(sys.argv) != 2):
        sys.exit('need the name of an image file')
    addText(sys.argv[1])
```



Datasets

MNIST digits





CIFAR-10 (http://www.cs.toronto.edu/~kriz/cifar.html)

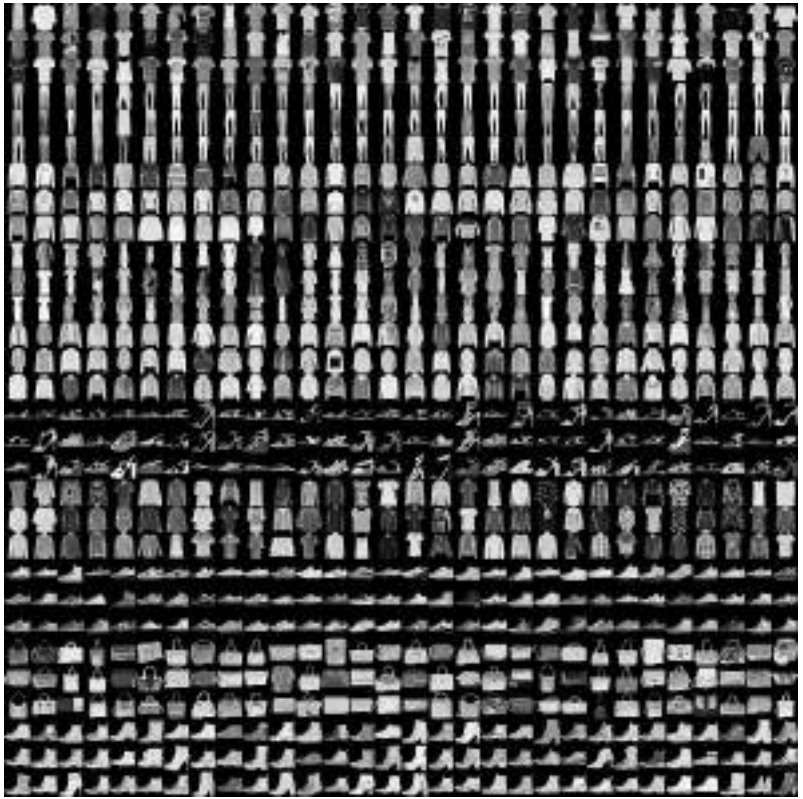
- Canadian Institute For Advanced Research
- 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck    
- 32 x 32 color images, 6,000 images per class
- CIFAR-100 dataset: 100 classes, organized into 20 superclasses, such as people, tree, fruit, insects, …

IMDB-WIKI (https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/)

- 523,051 face images
- date of birth, year when photo was taken, name …

Fashion MNIST (https://github.com/zalandoresearch/fashion-mnist)

- 10 classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot
- 60,000 training and 10,000 test images
- 28 x 28 gray-level



Cityscapes (https://www.cityscapes-dataset.com/dataset-overview/)

- semantic segmentation for vehicles and people
- 25,000 images with GPS coordinates

DAVIS (https://davischallenge.org/)

Densely Annotated VIdeo Segmentation

MOT Challenge (https://motchallenge.net/)

- Multiple Object Tracking (video data)
- purpose: object detection, pedestrian detection, 3D reconstruction, optical flow, single-object short-term tracking, and stereo estimation