FOUNDATIONS AND INTRODUCTION TO COMPUTER VISION

LEGAL NOTICES AND DISCLAIMERS

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

This sample source code is released under the Intel Sample Source Code License Agreement.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.

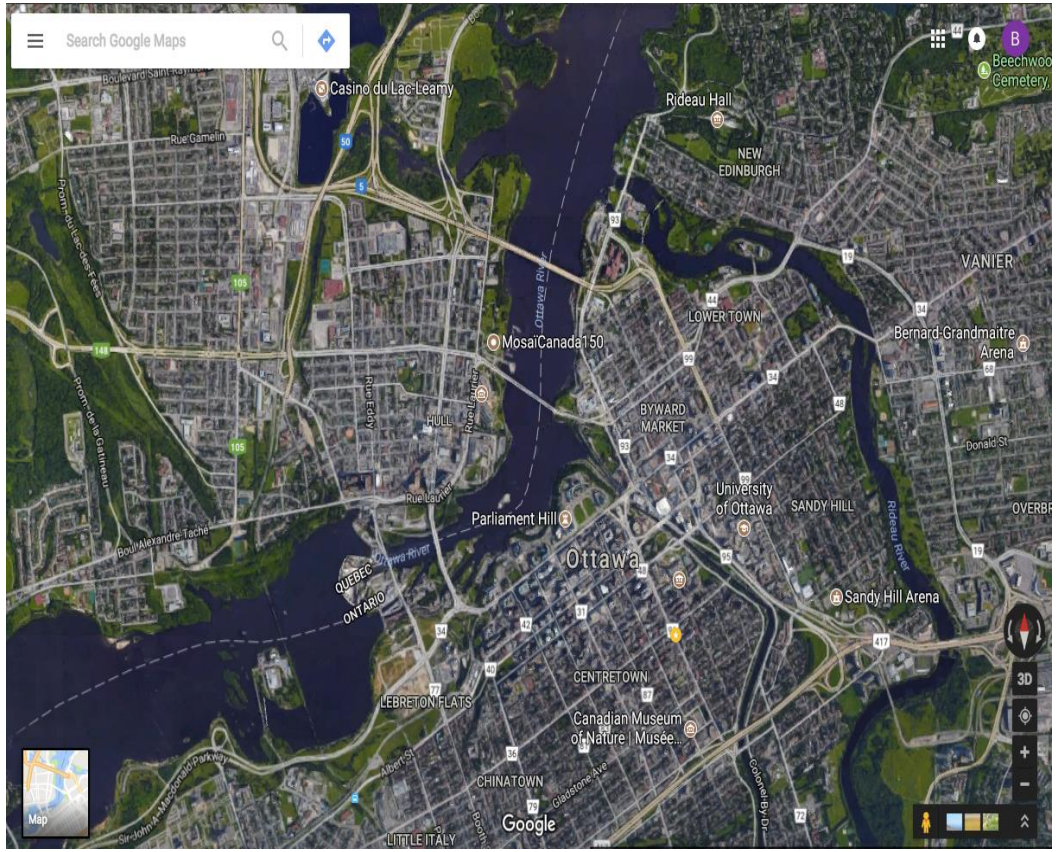
Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

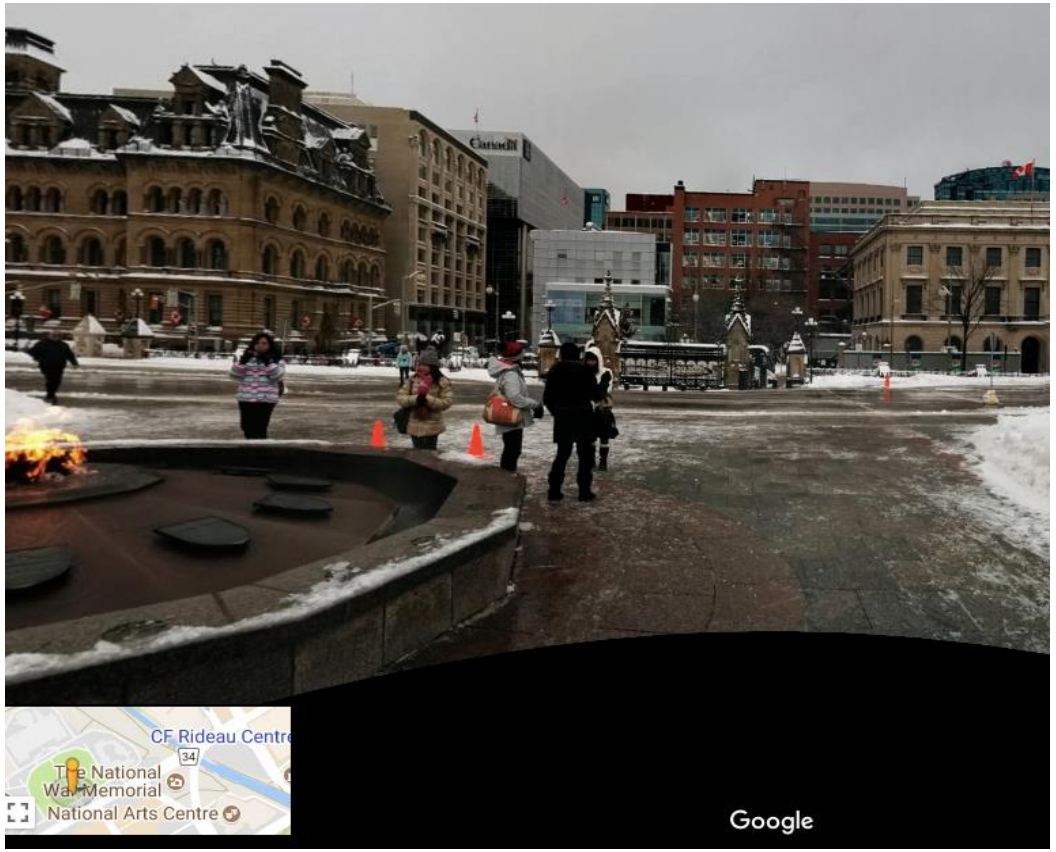
*Other names and brands may be claimed as the property of others.

WHAT IS COMPUTER VISION?

COMPUTER VISION INCLUDES...

Stitching together Google maps and street view





COMPUTER VISION INCLUDES...

Self-driving vehicles

Image from Wikimedia commons: https://upload.wikimedia.org/wikipedia/commons/1/1b/Google%27s_Lexus_RX_450h_Self-Driving_Car.jpg

COMPUTER VISION INCLUDES...

Interpretation of the visual world for robotics

Image from Wikimedia commons: https://commons.wikimedia.org/wiki/File:US_Navy_110817-N-PO203-027_Graduate_Students_prepared_autonomous_robots,_including_the_Office_of_Naval_Research_(ONR)-funded_shipboard_autonomous_f.jpg

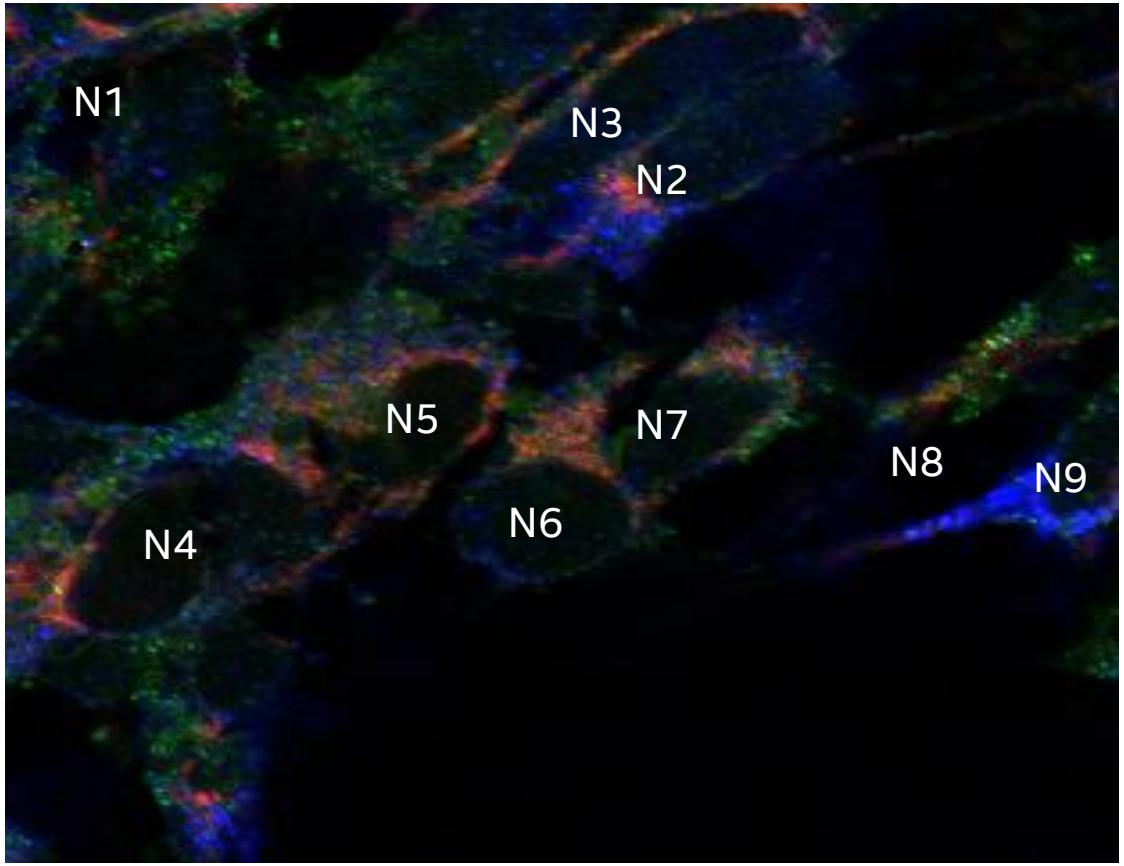
COMPUTER VISION INCLUDES...

Automated inspection of mass goods for industry

Image from Wikimedia commons: https://upload.wikimedia.org/wikipedia/commons/e/eb/Manufacturing_line.jpg

COMPUTER VISION INCLUDES...

Automatically labeling images



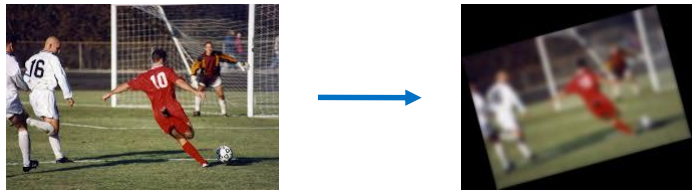
COMPUTER VISION INCLUDES...

Motion tracking of objects and people

Image from Wikimedia commons: https://commons.wikimedia.org/wiki/File:Silhouette_tracking.PNG

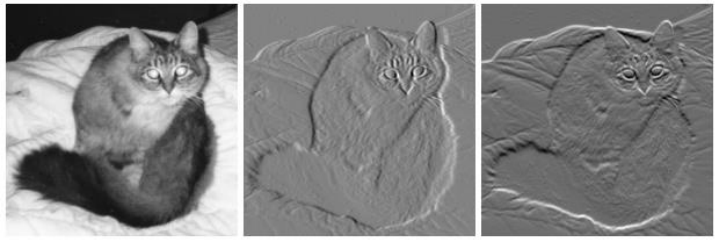
MANY TECHNIQUES NEEDED TO SOLVE THESE PROBLEMS

Solving real-world problems involves combining these techniques in a myriad of ways



Lesson 2: Image Processing

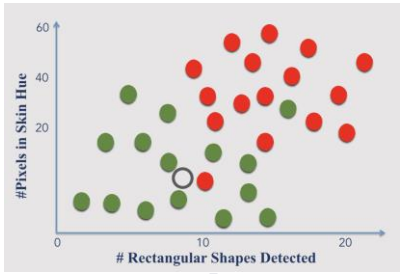
Lesson 5: Contours and Matching



Lesson 3: Image Transformations I

Lesson 6: Image Features

Lesson 4: Image Transformations II



Lesson 7: Machine Learning

IMAGENET* CHALLENGE (ILSVRC)

ImageNet* Large Scale Visual Recognition Challenge (ILSVRC) goals:

Develop best algorithm for large scale…

- Object detection
- Image classification
- Indexing for retrieval
- Indexing for annotation

As of 2017, computer algorithms showed 97.3% accuracy

Researchers use the ImageNet database to develop their neural networks for recognition tasks and then optimize their neural networks for their specific datasets (often much smaller and less optimal for learning model development)

COMPONENTS OF VISION SYSTEMS ARCHITECTURE

Image Capture

1. Characteristics of the physical world
2. Detection equipment
3. Image acquisition

Raw Data

1. Grayscale
2. RSB

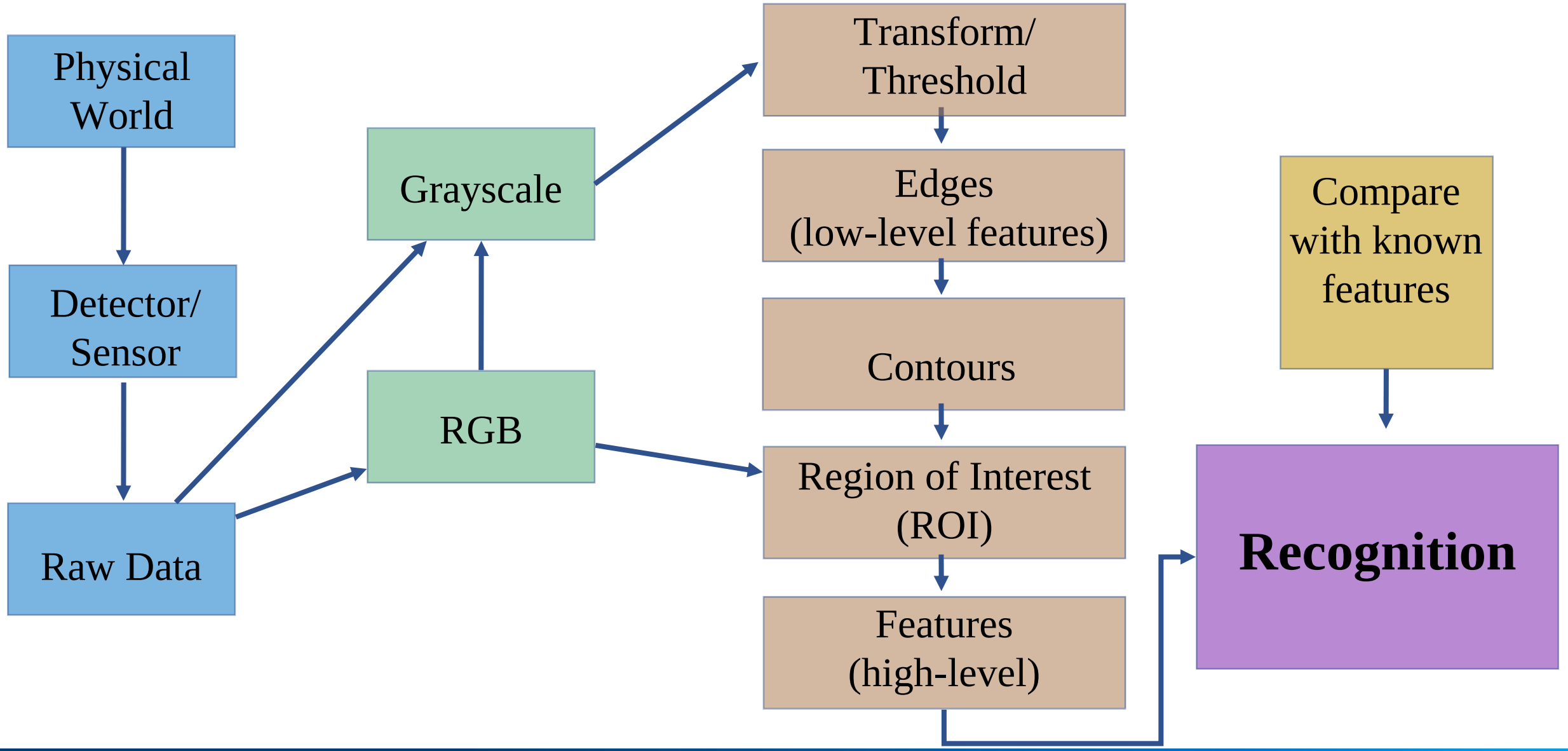
Preprocessing

1. Lower-level feature extraction

Processing

1. Detection and segmentation of image components
2. High-level feature extraction for object recognition
3. High-level processing
4. Decision making

VISUAL SYSTEM ARCHITECTURE



CHARACTERISTICS OF THE PHYSICAL WORLD

What light is getting to the detection equipment

- Color
- Brightness
- Intensity
- Light scatter

DETECTION EQUIPMENT

Physics of image formation: How light gets to the camera/sensor and how that sensor is activated

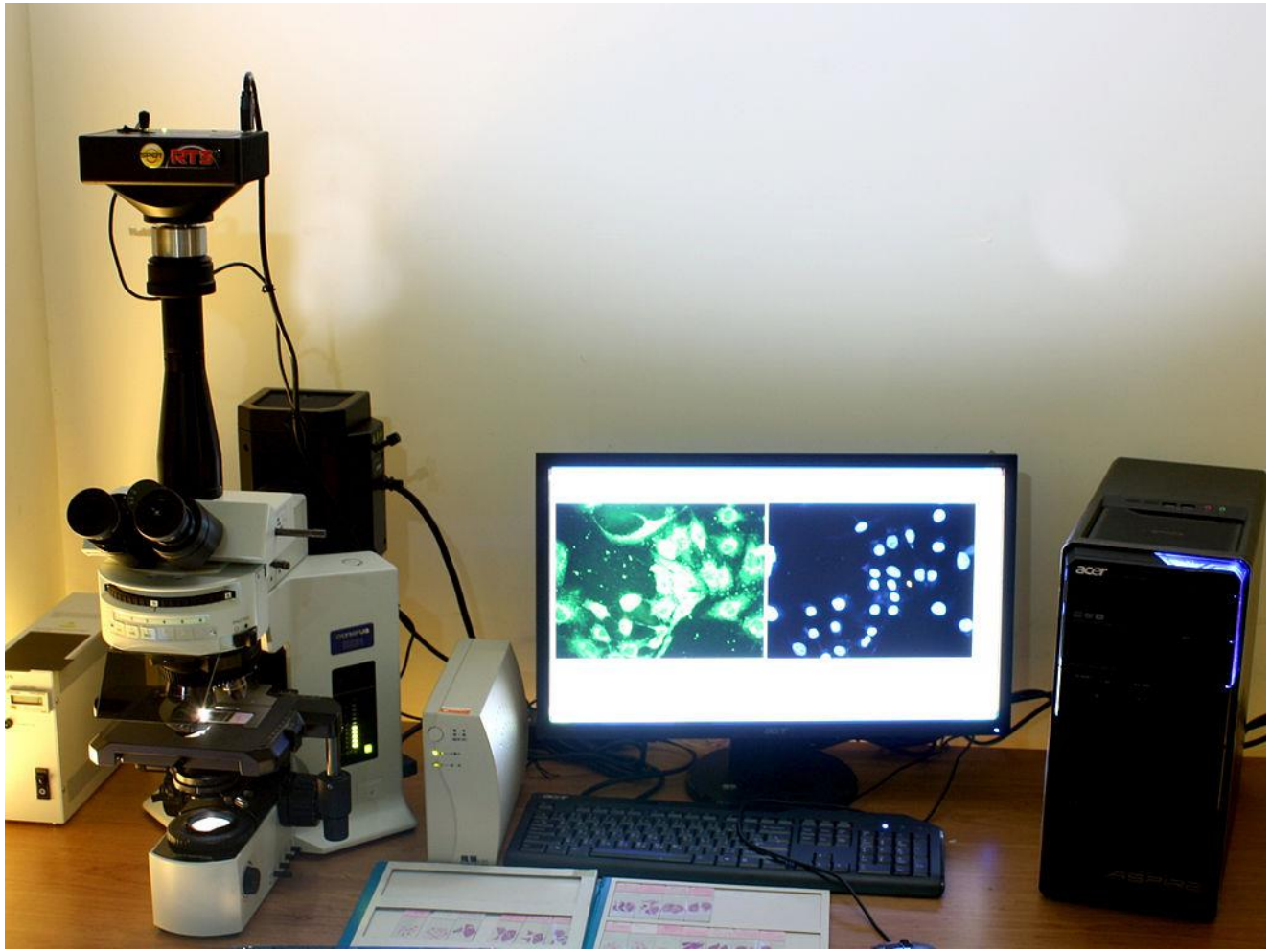
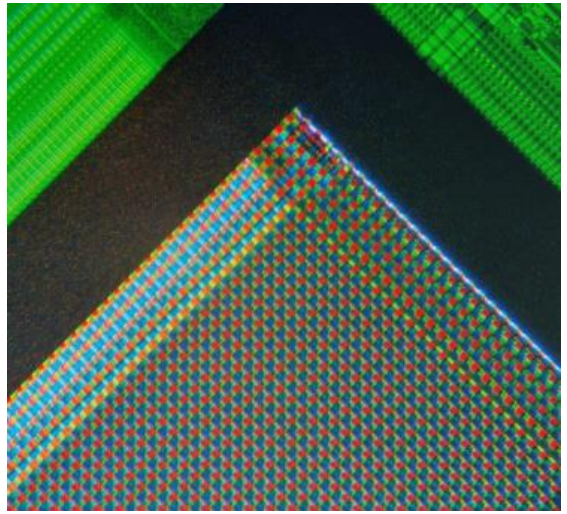


Image from wikimedia commons: https://commons.wikimedia.org/wiki/Category:Fluorescence_microscopes#/media/File:Olympus_Research_microscope_BX51.jpg

IMAGE ACQUISITION

Sensors turn light into electrical signals, or data.



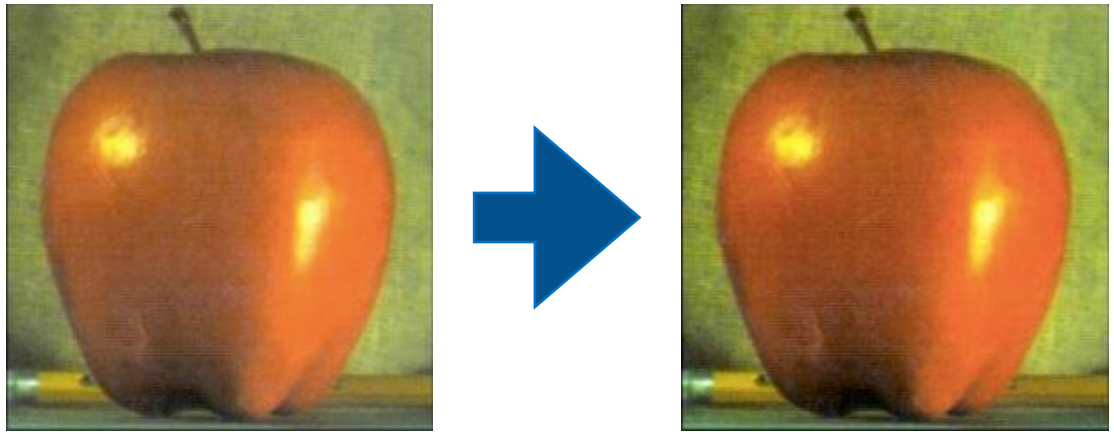
Images are colorized either by an additive process (in the case of RGB) or a subtractive process (in the case of CMYK, which is used in printers).

Image source: https://en.wikipedia.org/wiki/Image_sensor
Content source: http://www.shortcourses.com/sensors/sensors1-9.html

PREPROCESSING

Modern cameras use software to perform many preprocessing steps on images captured:

- Sampling data received to create image of appropriate resolution
- Denoising to increase sharpness
- Correcting contrast to increase clarity
- Scale correction



FEATURE EXTRACTION

Lines

Edges

Ridge

Corners

Blobs

Points

Texture/motion related

DETECTION AND SEGMENTATION OF IMAGE COMPONENTS

Define region of interest ("ROI", area with object of interest)

Scene segmentation - divide image into objects at the pixel level

HIGH-LEVEL PROCESSING

Counting and identifying attributes of objects of a type

Using object size, object position, or other feature and assigning these to specific categories

Counting attributes of specific object types

- Object size/position, objs → categories

Example: CV system could identify that

- There are 20 birds in this image
  - 5 ducks
  - 15 pigeons

Image source: https://upload.wikimedia.org/wikipedia/commons/e/ec/Birds%2C_Regent%27s_Park%2C_London_-_DSCF0446.JPG

HIGH-LEVEL PROCESSING

"Image understanding" (wikipedia)

Low level includes image primitives such as edges, texture elements, or regions

Intermediate level includes boundaries, surfaces, and volumes

High level includes objects, scenes, or events

LEARNING SYSTEM is implemented here

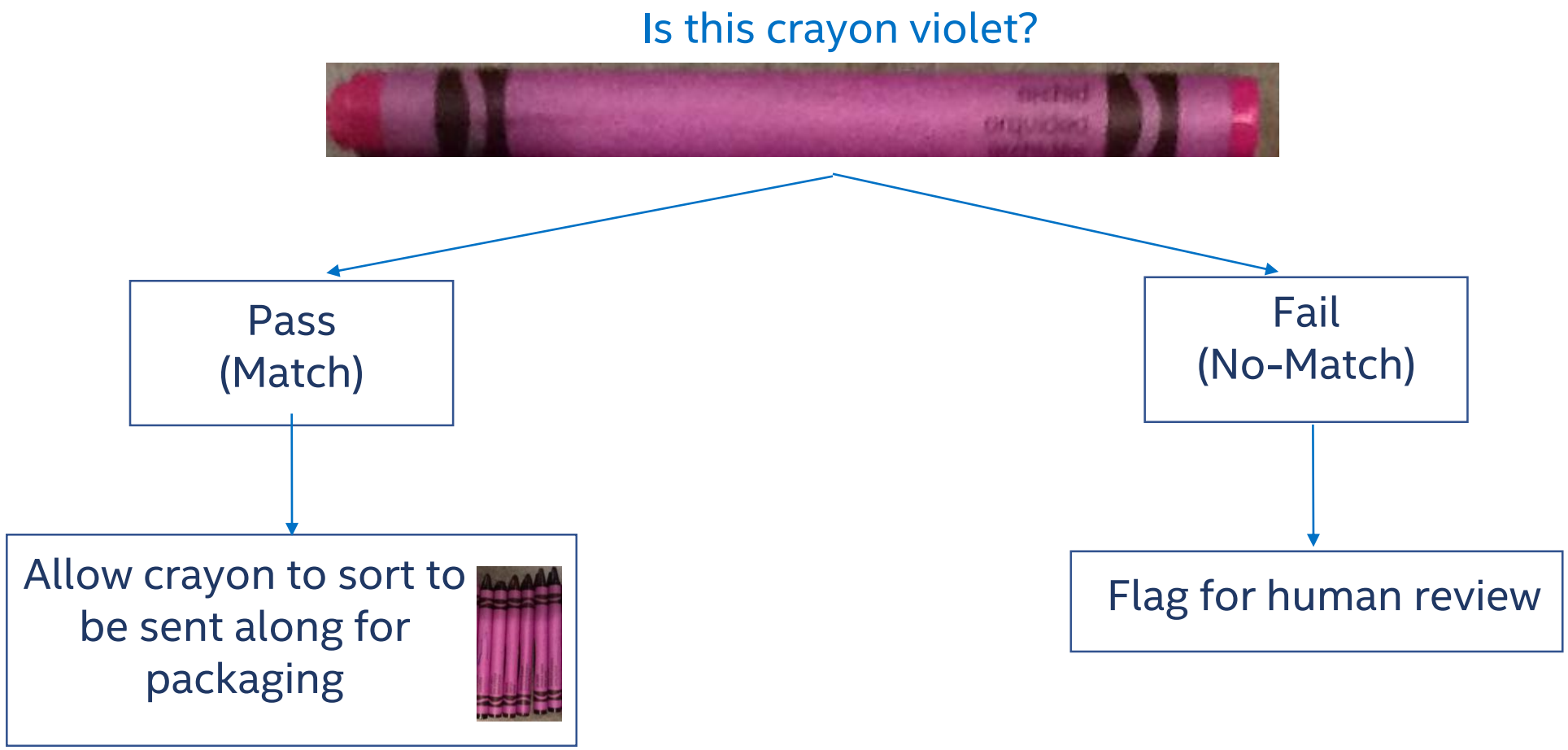
Examples

- Intermediate-level learning: "This image has grass, water, and concrete surfaces."
- High-level learning: "Birds are sitting near a pond in Regent's Park, London."

Image source: https://upload.wikimedia.org/wikipedia/commons/e/ec/Birds%2C_Regent%27s_Park%2C_London_-_DSCF0446.JPG

EXAMPLE: AUTOMATED ASSEMBLY

In this scenario, we have an automated assembly sorting violet crayons. The robot must identify whether the crayon is violet.



RELATED DISCIPLINES AND CONTEXT

Computer Graphics

- Takes structural input data to create a single 2D image

Image Processing

- Extracting data from an image and converting it to usable data for higher-level processing
- Point operators, pixel transforms, color transforms, compositing and matting, histogram equalization

Image Analysis

- Uses 2D images to produce a new 2D image (from image processing steps) or generate statistics from the original 2D image
- High-level processing and feature extraction

RELATED DISCIPLINES AND CONTEXT

Computer Vision

- Integrates multiple 2D images to generate 3D models of objects and scenes
- Optical character recognition, machine inspection, 3D model building, medical imaging, automotive safety, robotics

Machine Vision

- Uses computer graphics, image analysis, and computer vision tools as part of automated system
- QC, monitoring, robotics, and so on

FOUR MAJOR TECHNOLOGIES

OpenCV

Python*

NumPy

matplotlib*

OPENCV

WHAT IS OPENCV?

Written in C++ using Standard Template Library (STL)

BSD license

Bindings in several languages

- c
- c++
- Java*
- Python*
- Matlab*

Supports several platforms

- Windows*
- Linux*
- macOS*
- Android*
- iOS*

Open source Computer Vision and Machine Learning library

OPENCV ALGORITHMS ARE IMPLEMENTED TO...

- Detect and recognize faces
- Identify objects
- Classify human actions in videos
- Track camera movements
- Track moving objects
- Extract 3D models of objects
- Produce 3D point clouds from stereo cameras
- Stitch images together to produce a high resolution image of an entire scene
- Find similar images from an image database
- Remove red eyes from images taken using flash
- Follow eye movements
- Recognize scenery and establish markers to overlay it with augmented reality

MAJOR COMPONENTS OF OPENCV

Core

- Is fundamental image as matrix computations
- NumPy will do most of core computations for us

imgproc

- Great fundamental routines

highgui

- Cross-platform rendering and image interaction

Other

- Primary (open source) modules for tasks: object detection, deep neural networks, and machine learning, GPU-based calculation (Cuda*), video, and so on
- Contributed extra (non-free) and third-party modules

RELATIONSHIP BETWEEN OPENCV CORE AND NUMPY

Use opencv/modules/python/src2/cv2.cpp

To provide a NumPy interface to the raw C++ image structures

This lets us replace much of opencv2.core with generic (and short!) NumPy code

RELATIONSHIP BETWEEN OPENCV CORE AND NUMPY

C++ Code

```
void saturate_sv( IplImage* img ) {
  for( int y=0; y<img->height; y++ ) {
    uchar* ptr = (uchar*) (img->imageData + y * img->widthStep );
    for( int x=0; x<img->width; x++ ) {
      ptr[3*x+1] = 255;
      ptr[3*x+2] = 255;
    }
  } /* for */
} /* void */
```

Python* Code

```
def saturate_sv(img):
  H,S,V = [0,1,2]
  img[:,:,[S,V]] = 255
```

RELATIONSHIP BETWEEN OPENCV CORE AND NUMPY

C++ Code

```
IplImage* doCanny(IplImage* in,
                  double lowThresh,
                  double highThresh,
                  double aperture) {
    if(in->nChannels != 1)
      return(0);
    IplImage* out = cvCreateImage(cvSize(cvGetSize(in)),
                                  IPL_DEPTH_8U,
                                  1);
    cvCanny( in, out, lowThresh, highThresh, aperture );
    return( out );
}
```

Python* Code

```
def doCanny(img, lowTh, highTh, aperature):
  if len(img.shape) != 1:
    return 0
  return cv2.Canny(img, 100, 200)
```

PYTHON*

ABOUT PYTHON*

License

Python Software Foundation (PSF) license (GNU General Public License (GPL) compatible)

Design Principles (Zen of Python*)

Balance consistency, practicality, and readability

Active Communities

Commercial Support

ABOUT PYTHON*

Ease of Learning and Reading

Sparse syntax

Interactive Prompt

Rapid Development Cycle

Batteries Included

For *many* common tasks

Third-party

Modules and packages for just about everything else!

NUMPY

WHAT IS NUMPY?

- Foundation of numerical computing (number crunching) in Python*
- Basis of sciPY/ scikit-* and so on
- Used for memory storage of OpenCV!
- Roots trace to mid-90s with "Numeric" and then NumArray
- NumPy 1.0 was released in 2006
- Recent stable version of NumPy was released 29 September 2017; 1.13.3

NDARRAYS: PRIMARY DATA STRUCTURE

Using ndarrays, we code vectorized operations

a*b instead of explicit Python* looping

But, under the hood, ndarray provides

- Fixed memory layout
- Fixed datatype
- Like a C-array on the heap
- Raw memory, therefore not the high-level language overhead of Python
- Localized memory, which takes advantage of (spatial) caching locality
- We perform vectorized operations at the CPU level

OPERATIONS WITH ARRAYS

NumPy

- Built-in routines for numerical mathematics, statistics, and matrix operations

Broadcasting

- Matching elements between arrays containing shapes that are not identical
- Repeating, without copying, smaller dimensions to larger

OPERATIONS WITH ARRAYS

ufunc

Fast, element-wise array operation

- Lots of wrapping gets us to C code
  - numpy/core/src/umath/funcs.inc.src#L111
- Don't believe it, look here...
  - numpy/core/code_generators/generate_umath.py

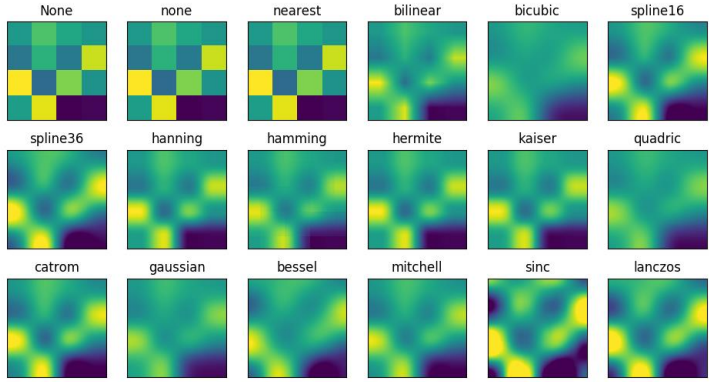
Linking

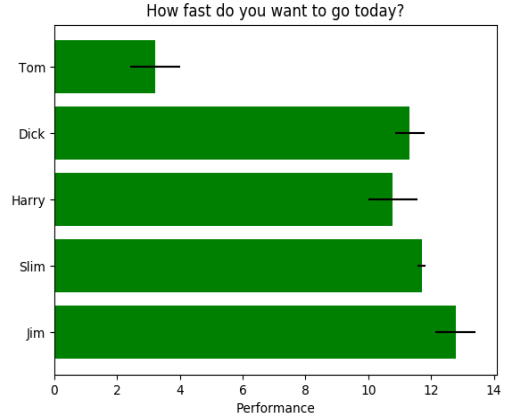
To fast C and Fortran code for matrix operations

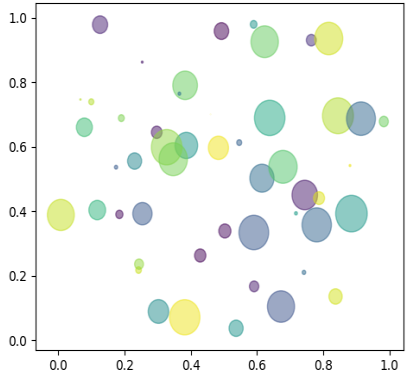
- dot -> BLAS/ATLAS/MKL code

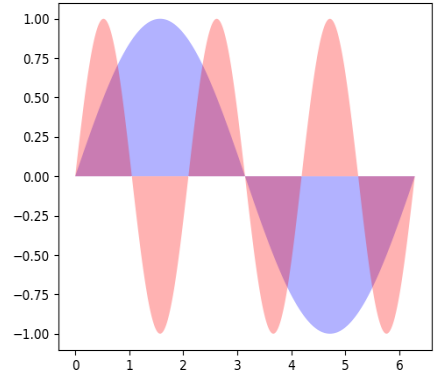
MATPLOTLIB*

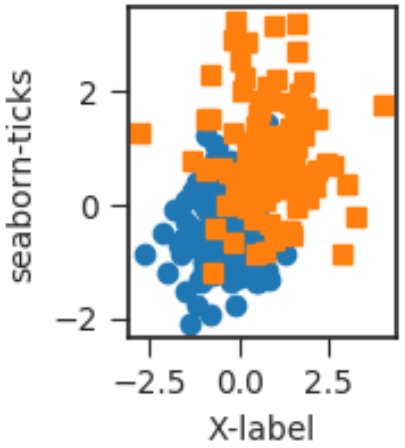
WHAT IS MATPLOTLIB*?

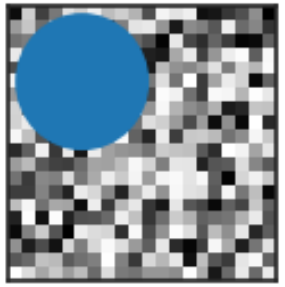


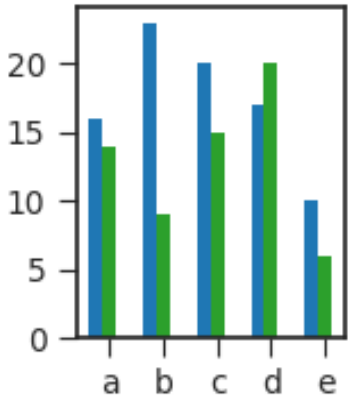


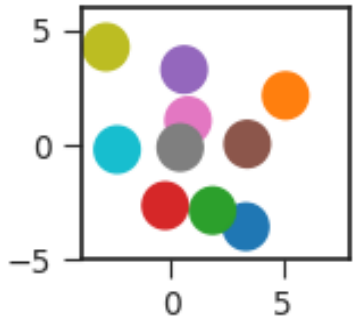


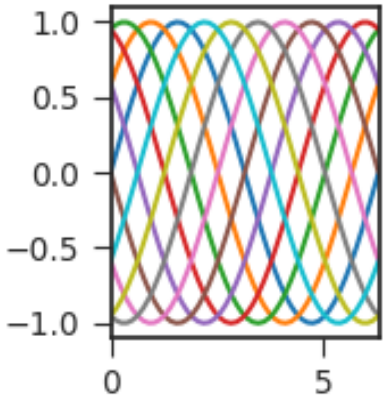


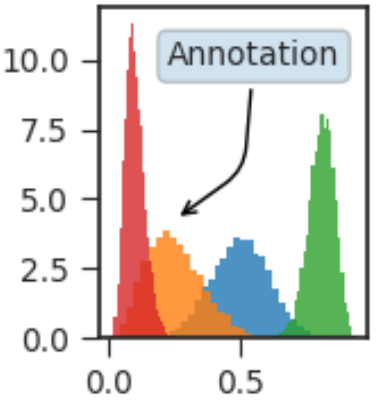












ABOUT MATPLOTLIB*

Originally designed by John D. Hunter to emulate Matlab*

Python* Software Foundation (PSF); BSD-style license

First released in 2003

Recent stable release of 2.1.0 on October 8, 2017

WHAT IS MATPLOTLIB*?

Primarily a library for 2D plotting of:

- Array data in Python*
- Statistical plots
- Images

Primarily written in Python (with NumPy) and some extensions

Additional toolkits (for example, mplot3d for 3D plots)

Imported separately

Third-party packages (for example, seaborn*) built on Matplotlib

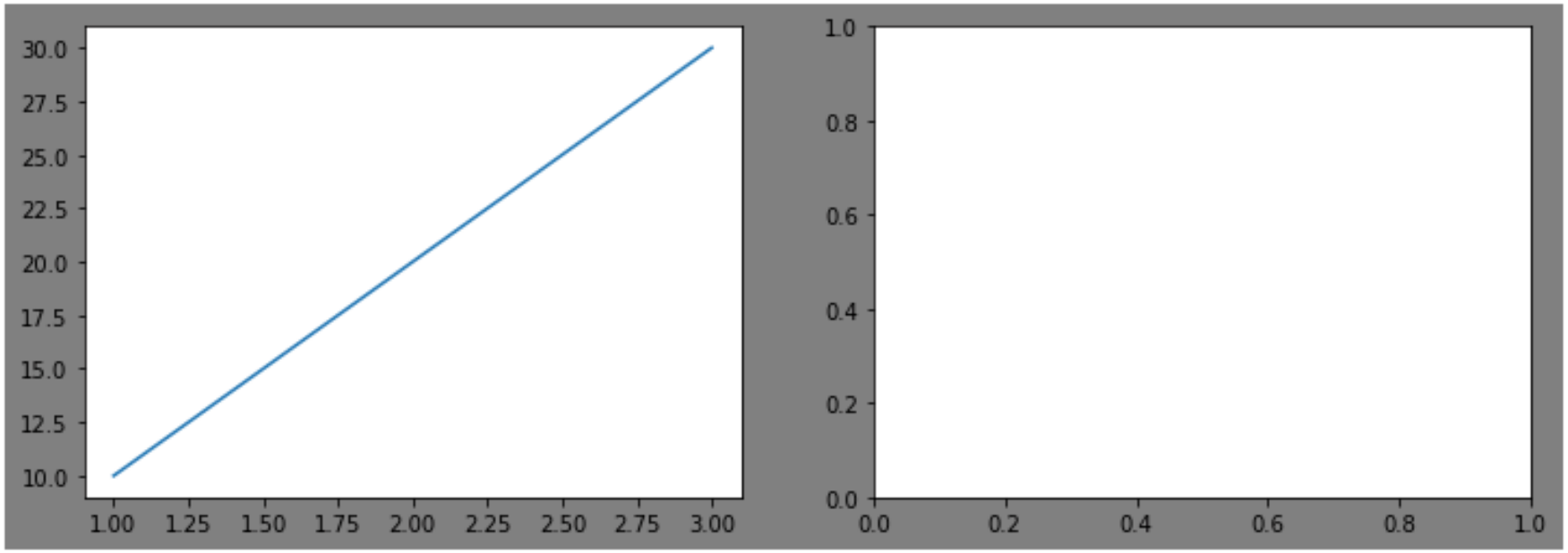
http://matplotlib.org/thirdpartypackages/index.html

USING FIGURES AND AXES IN MATPLOTLIB*

Figure is "whole graphic"

Axes are individual plots

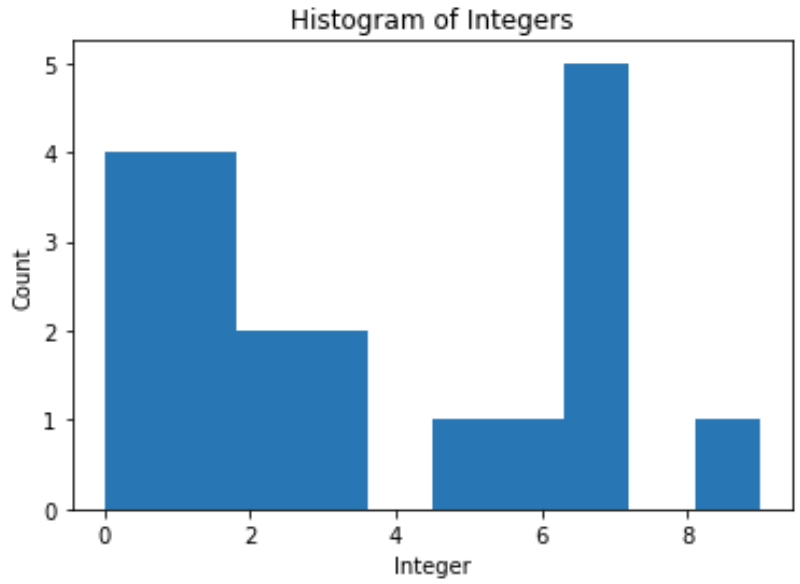
Graphics with figures/axes (color differently: Figure(facecolor='red') can have axes embedded within axes (for example, zoom-in)



TYPICAL USE PATTERN

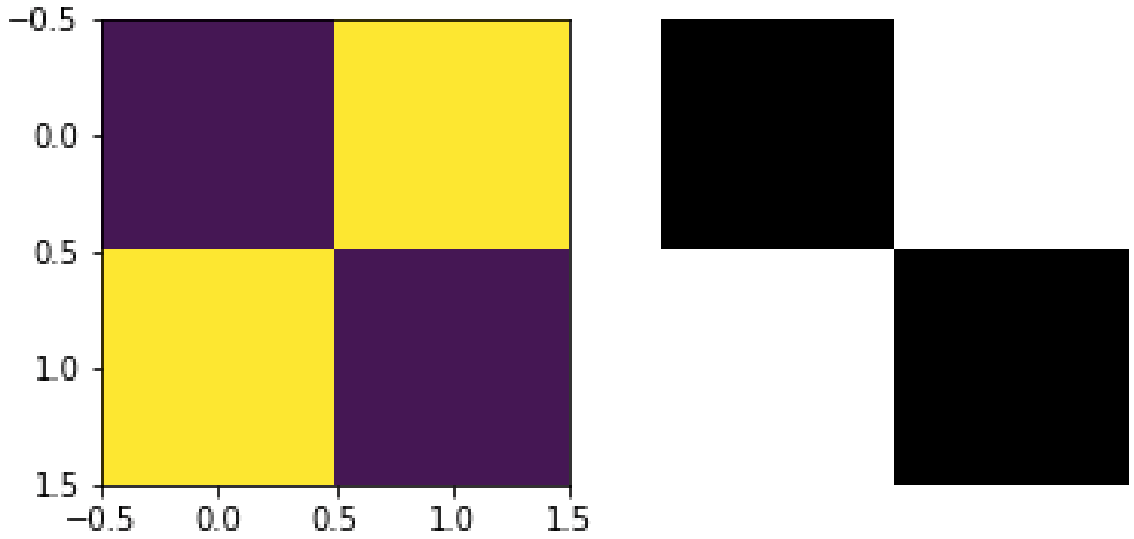
```
ax = plt.gca()
arr = npr.randint(0,10,20)
ax.hist(arr, bins=10)

ax.set_title("Histogram of Integers")
ax.set_xlabel("Integer")
ax.set_ylabel("Count")

import collections as co
print(co.Counter(arr)) # pure python counts of occurances
```

```
Counter({7: 5, 1: 4, 0: 4, 2: 2, 3: 2, 5: 1, 9: 1, 6: 1})
```



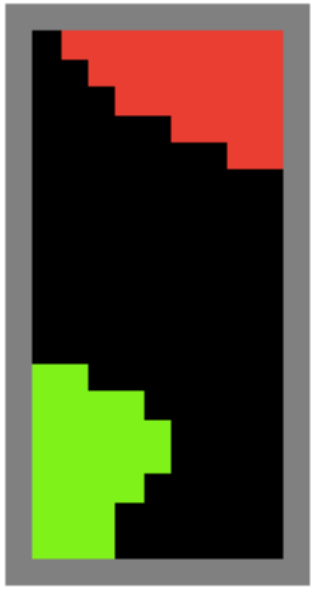
TYPICAL USE PATTERN

```
fig, axes = plt.subplots(1,2)

arr = np.array([[0,1],
                [1,0]], dtype=np.float64)
axes[0].imshow(arr)

axes[1].imshow(arr, cmap='gray')
axes[1].axis('off');
```

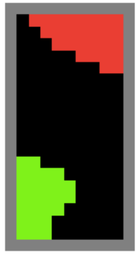


IMAGES AS ARRAYS

PIXELS



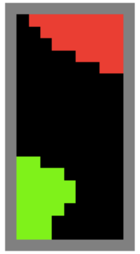
PIXELS: RED



```
R,G,B = 0,1,2
region[:,:,R]
```

```
array([[128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128],
       [128,   0, 255, 255, 255, 255, 255, 255, 255, 255, 128],
       [128,   0,   0, 255, 255, 255, 255, 255, 255, 255, 128],
       [128,   0,   0,   0, 255, 255, 255, 255, 255, 255, 128],
       [128,   0,   0,   0,   0,   0, 255, 255, 255, 255, 128],
       [128,   0,   0,   0,   0,   0,   0,   0, 255, 255, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128]], dtype=uint8)
```

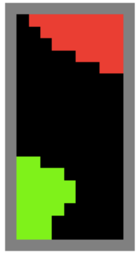
PIXELS: GREEN



```
region[:,:,G]
```

```
array([[128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128, 255, 255,   0,   0,   0,   0,   0,   0,   0, 128],
       [128, 255, 255, 255, 255,   0,   0,   0,   0,   0, 128],
       [128, 255, 255, 255, 255, 255,   0,   0,   0,   0, 128],
       [128, 255, 255, 255, 255, 255,   0,   0,   0,   0, 128],
       [128, 255, 255, 255, 255,   0,   0,   0,   0,   0, 128],
       [128, 255, 255, 255,   0,   0,   0,   0,   0,   0, 128],
       [128, 255, 255, 255,   0,   0,   0,   0,   0,   0, 128],
       [128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128]], dtype=uint8)
```

PIXELS: BLUE



```
region[:,:,B]
```

```
array([[128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128,   0,   0,   0,   0,   0,   0,   0,   0,   0, 128],
       [128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128]], dtype=uint8)
```

ARRAY OPERATIONS AND IMAGE ANALOGUES: CONTRAST AND BRIGHTNESS

```
frog = my_read('data/frog.jpg')

fig, axes = plt.subplots(1,3, figsize=(12,3))

my_show(axes[0], frog)
my_show(axes[1], np.clip(.5 * frog, 0, 255).astype(np.uint8))
my_show(axes[2], np.clip(.5 * frog + 50, 0, 255).astype(np.uint8))

axes[0].set_title("(I) Original")
axes[1].set_title("(II) Reduced Contrast")
axes[2].set_title("Increased Brightness on II");
```

(I) Original

(II) Reduced Contrast

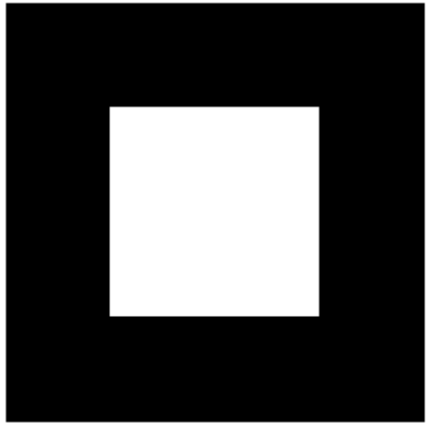
Increased Brightness on II

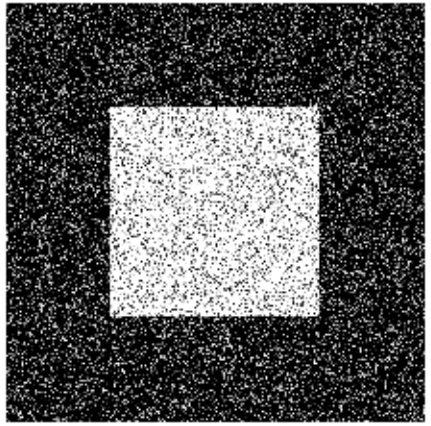
ARRAY OPERATIONS AND IMAGE ANALOGUES: BLENDING

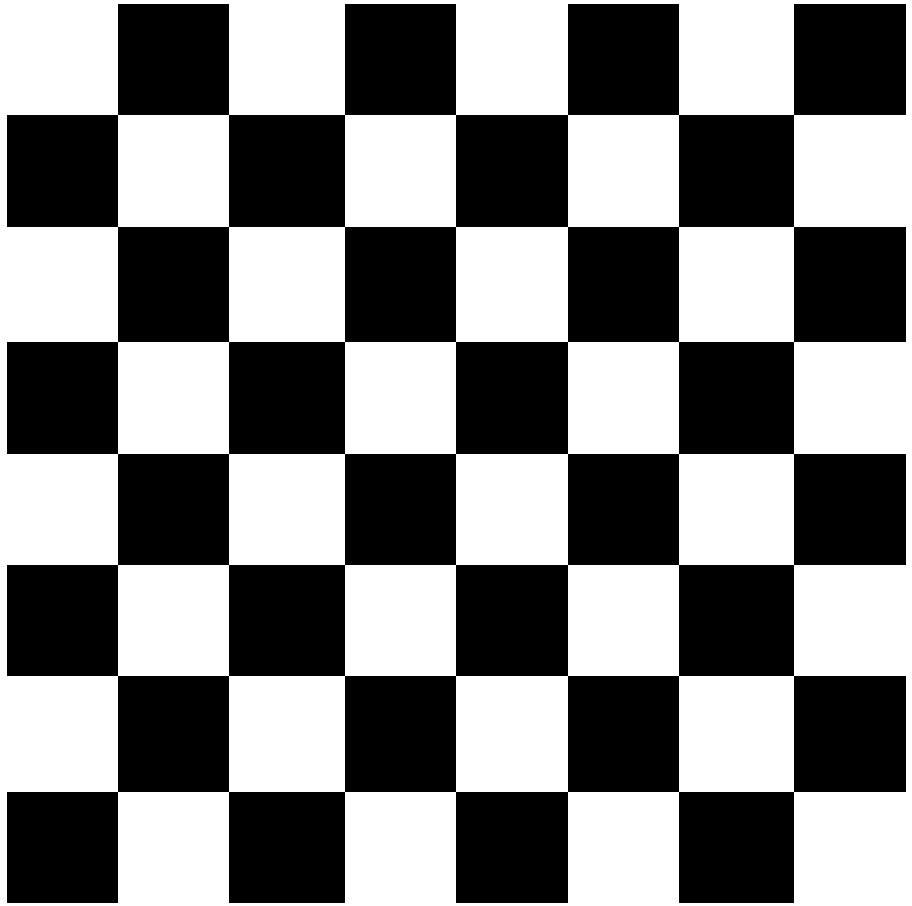
```
ml    = my_read('data/ml.png')

fig, axes = plt.subplots(1,3, figsize=(12,3))

min_r, min_c = (min(ml.shape[0], frog.shape[0]),
                min(ml.shape[1], frog.shape[1]))
min_slice = (slice(0, min_r), slice(0,min_c))

my_show(axes[0], frog)
my_show(axes[1], ml)
my_show(axes[2], (.3*frog[min_slice] + .7*ml[min_slice]).astype(np.uint8))

axes[2].set_title(".3 Frog + .7 Robot");
```

.3 Frog + .7 Robot

TEST IMAGES







RGB HISTOGRAMS

```
color_to_index = {"R":0, "G":1, "B":2}  # map strings to appropriate index in

fig, axes = plt.subplots(1,3,figsize=(12,3), sharey=True)
for ax, color in zip(axes, color_to_index):
    c = color_to_index[color]
    this_channel = messi_rgb[:,:,c].ravel() # 1D view without copying

    ax.hist(this_channel, 256, normed=True)
    ax.set_title("Histogram for {}".format(color))
```