Introduction to Computer Vision

## Objectives

* Understand what computer vision is
* Apply computer vision algorithms with Python and OpenCV
* Create custom classifiers.
* Build web apps that classify images

## Not Objectives

* How computer vision works
* How neural networks and deep learning works
* Maths and statistics

### What is Computer Vision?

* As humans we can instantly recognize the contents of an images
* Computers cannot
* Computer vision provides computers the ability to see and understand images

### Why is Computer Vision used?

Like all technologies it is used to:

* Make something faster
* Make something cheaper
* Make something automated
* Make something easy
* Make something convenient
* Make something scalable

Applications include self-driving cars, processing images involved in checking things such as rock samples, rust on electricity pylons etc using ‘classifiers’ (metal classifier, rust classifier). Sifting through videos using keywords based on objects in each scene (security footage etc).

**Recent Research in Computer Vision**

* Image detection –
* Image conversion/translation – horse to zebra, winter to summer
* Motion transfer – transferring motion from one target to another

**Brainstorming Applications**

* Start with a problem
* How to identify problems?
  + What are time-consuming tasks that are visual?

### What is a Digital Image?

Its essentially a rectangular array of numbers.

When zoomed in, a picture is just pixels with each pixel represented by a value of 0 – 255 in one of three colours Red, Green or Blue (channels)

Less values = more contrast

Each pixel value comes from a grid of sensors. The picture is the quantized samples obtained from an original object.

Image maps highlight object in the picture by defining pixels of that object as one and everything else as zero.

A video sequence is simply a sequence of images.

**Pillow**

Pillow is a python library for working with images.

To work with Pillow

1. from PIL import Image

2. # image can be rendered using

3. image = Image.open(my\_image)

4. # image can also be rendered using matplotlib

5. import matplotlib.pyplot as plt

6. plt.imshow(my\_image)

7. image.format:PNG

1. # the format attribute is the image format

2. # the size attribute is the size

3. image.size: -> (512,512)

4. # the mode attribute is the colour space

5. image.mode: -> (RGB)

6. # the ImageOps module contains methods related to image manipulation

7. image\_grey = ImageOps.grayscale(image)

8. # image mode L is luminence

9. image\_grey.mode: -> (L)

10. # images can be saved and converted to JPG

11. image\_grey.save(“lenna.jpg”)

12. # quantize can be used to quantize the image

13. image\_grey.quantize(2)

14. # the colour channels can be split

15. r,g,b = image.split()

16.

Plotting the channels as greyscale images shows that the colour shows up as higher values (aka, if red values high, on greyscale, red areas are brighter)

Images can be converted to numpy arrays using the numpy constructor

1. array = np.array(image)

OpenCV also creates these arrays.

It has more functionality than PIL and is used for computer vision

1. # to import and read img

2. import cv2

3. image = cv2.imread(my\_img)

4. # this converts the image straight into numpy

5. type(image): -> numppy.ndarray

6. # shape of the image

7. image.shape: -> (512,512,3)

8. # image can be plotted using plt however the colours will be warped

9. plt.show(image)

The colour will be warped because of the arrangement of the different channels and how plt interprets them.

In PIL the channel order is RGB, in Cv2 they are BGR

The colour code can be changed

1. new\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

2. # !! slices can be used to obtain the different colour channels

3. b,g,r = img[:,:,0], img[:,:,1], img[:,:,2]

4.

The first two slices select the horizontal and vertical, the final, selects the channel