Quick Quiz

https://www.pollev.com/ivorsimpson490

University of Sussex

Lecture 2: Python for Computer Vision and the digital representation of images

Ivor Simpson

University of Sussex

Today's Outline

- Python overview.
- Useful packages for CV/ML
- How are images represented and how can they be resampled
- How are digital photographs acquired

Learning Outcomes

- Understand how Python can be used for Computer Vision
- Be able to write some Python code to perform simple manipulation and examination of images.

Why Python?

- Python is probably the most popular language in computer vision/machine learning.
- Easy to do rapid prototyping, and can be very efficient (if written correctly)
- Fantastic library support for computer vision and machine learning
- Open source
- Easy to interface with other languages, e.g. C++.
- Acceleration can be achieved using packages like Numba or on GPU using TensorFlow or PyTorch

Sounds great, are they any drawbacks?

- Things like for loops can be very slow - interpreted rather than compiled. Efficiency can be improved though (more on that later)
- Can be tricky to build very large projects. Interpreter doesn't enforce best practice
- Different Python and Python library versions (and subversions!) are not necessarily compatible.
- Package dependency management can be complicated.

Running Python

There's a few options:

- Using Colaboratory - you can load most standard packages without any trouble.
- Install Python directly on your own machine, or use Anaconda (or something similar). The lab machines do have Python, but don't rely on the correct packages being installed.
- https://docs.python.org/3/tutorial/venv.html allow you to have different python environments with different packages installed.
- https://www.jetbrains.com/pycharm/ is a great Python IDE, free for academic use.

How to write Python

There's a few options:

- People often start by using ipython (interactive python), which is what Jupyter/Colab use in the background.
  - This gives a mechanism for inspecting what happens line by line.
  - Although this is a useful format for interaction, it's not great for code reuse, writing unit tests etc.
- For things you might run a few times, you could copy the notebook code into a script.
- As you write more reusable and tested code you should refactor it into classes and modules.
- This prevent the issue of running cells out of order! or having to copy bugfixes to all code instances.

Useful Packages

- packages are installed using pip
- numpy For basic numerical and linear algebra functionality
- scipy For a variety of scientific libraries
- matplotlib For all your data plotting needs
- opencv Very well used computer vision library
- scikit-learn for some standard implementations of ML models.
- Deep learning packages like TensorFlow or PyTorch.

Python's basic Data Structures

https://www.pollev.com/ivorsimpson490

- The most important data structure of us is an array, which is an n-dimensional set of numbers of a given datatype (e.g. uint8 or float32)
  - These can be sliced in many useful ways
  - negative indices start from the back of the list, e.g. $a[-1] == 2$
  - you can reverse the contents by $a[:: -1]$
- Lists and tuples can all store elements of any type
  - lists $a = [1, 2]$ are mutable containers indexed with integers.
  - tuples $b = (1, 2)$ are immutable containers indexed with integers.

Python's basic Data Structures

- dictionaries give you a mapping between keys and values
- They are commonly initialised like this
  $a = \{'first' : 1, 'second' : 2\}$
- They are indexed by $a['first']$ (throws an error if doesn't exist) or $a.get('first')$ returns None if doesn't exist
- They are mutable
- The keys and values can be any mixture of types as long as the key is hashable.

University of Sussex

Useful things to know about

- iterators and the use of enumerate
- zip
- asserts
- decorators

Building Arrays

numpy contains an array class to represent numerical data.

```
np.array([0.0, 1.0, 2.0], dtype=np.float32)
```

There are many ways to initialise them, to make an array with dimensions $10 \times 6 \times 3$:

```
np.ones((10,6,3), dtype=np.int)
```

or with random numbers

```
np.random.randn(10,6,3)
```

Operations on Arrays

```
A = np.array(...)
```

Multiply the array contents in-place

```
A *= 2.0
```

Return a new array which is A * 2

```
B = A * 2
```

Returns the shape of the array as a tuple, e.g. (10, 6, 3)

```
A.shape
```

Operations on arrays

You can average over a particular array dimension:

```
np.mean(A, axis=0, keepdims=True)
```

where keepdims keeps it as a 3D array, so A.shape $= (1, 6, 3)$
Arrays can be multiplied together, elementwise:

```
A *A == np.square(A)
```

You can perform operations on arrays of different sizes (in some cases by broadcasting. In this case we can subtract the columnwise means by:

```
A - np.mean(A, axis=1, keepdims=True)
```

Broadcasting only works when the arrays dimension sizes match, or one of the dimensions is of size 1.

Slicing Examples

Arrays are very good at dealing with slicing operations, e.g., to choose every other row

```
A[0::2,:,:]
```

You can do assignment with these slices too

```
A[0::2,:,:] *= 2.0
```

You can reverse along a particular dimension, e.g. channels

```
A[:,:,::-1]
```

Linear Algebra Ops : Matrix Multiplication

You can also treat arrays as Matrices/Vectors

- either using function like np.matmul (matrix multiply)
- look in np.linalg for more functions
- You can also cast an array to always act like a matrix using np.matrix(A)

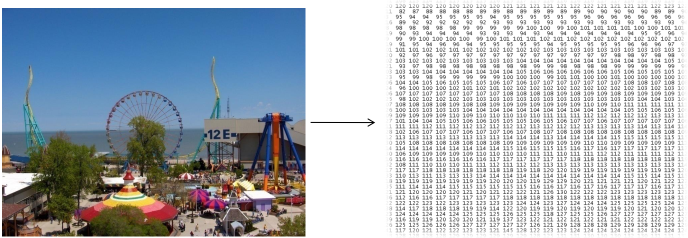
Writing Efficient Numpy

- Things like for loops can be very slow in Python as it's interpreted rather than compiled.
- Try and use built in vectorised operations to maintain efficiency
- or compile to C using Cython.

Finally, some images!

- How will we represent images?
  as arrays!
- But what do the numbers mean?

What the computer gets



Kristen Grauman

University of Sussex

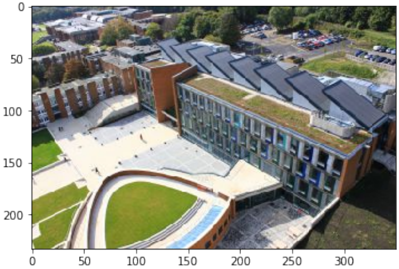
Key points on images as arrays

- Pixels (picture elements) describe the sampling of incoming light on a regular grid.
- The set of pixels that make an image are stored as an array.
- The numbers at each pixel represents the colour at that point
- The shape of the array is (height, width,number of colours)
- The image coordinate system normally starts with (0,0) as the top-left pixel and ends with (h-1,w-1) at the bottom right pixel.
- Typically the original signal information is quantised to 8-bits per channel.

Visualisation

Use matplotlib!

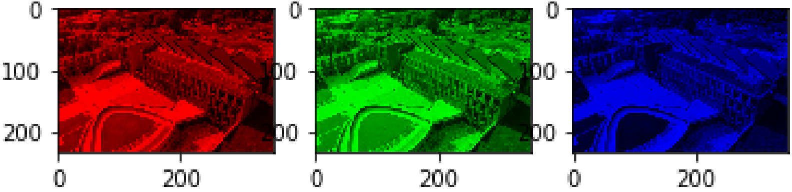
```
from matplotlib import pyplot as plt

plt.imshow(img, vmin=..., vmax=...)
plt.show()
```



simple notebook lab notebook

The data representation of images

Each channel (3rd array dimension) encodes a color



An easy (but inaccurate) way to make an image grayscale (i.e. just brightness) is to average the colour at each pixel.

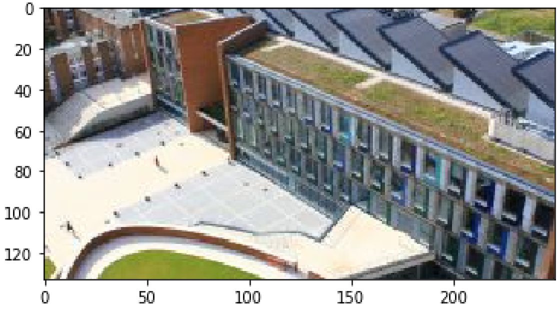
University of Sussex

Simple way to halve the resolution of an image

```
img[::2,::2,:]
```

How can we crop the center of an image

```
imgrgb[50:-50:,50:-50:,:]
```



Summary

- Python has lots of useful libraries for computer vision
- Check out the week 1 lab for some examples of how to perform simple image manipulation.
- Next time we'll still looking at some slightly more complex operations on images.

University of Sussex