Advanced Image Processing

Andrew Green

MANCHESTER 1824
The University of Manchester

The Christie NHS
NHS Foundation Trust

The Plan

- Recap
  - Loading data
  - Image processing
  - Image registration
  - Image display for fusion
  - Manual image fusion
- Loading DICOM into python
- Cost function
- Automatic image fusion
- Intro to the practical

Loading data

- Remember the key functions for loading data:
  - Images: `misc.imread`
  - CSV files: `np.loadtxt`
- Flatten colour images to make them black & white
  - Averages colour channels
  - Optional argument `flatten=True` to imread
- Specify delimiter for loadtxt
  - CSV files will be `delimiter=‘,’`

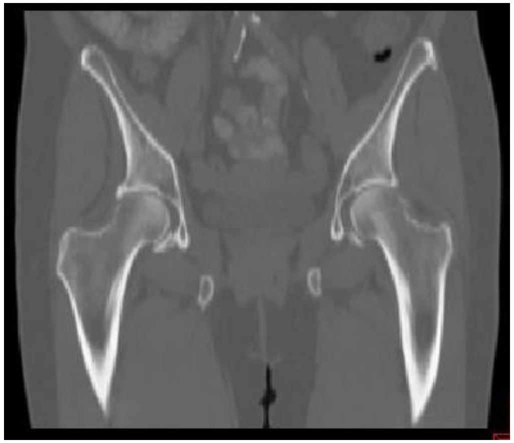
Intensity transformations:
Window/Level

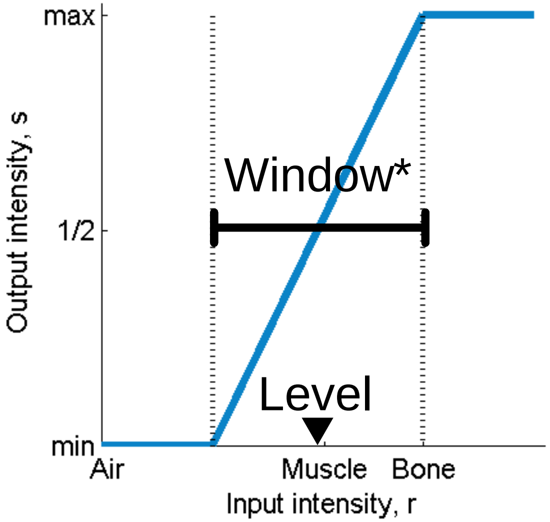
- Window/level can also be represented as intensity transformation → piece-wise functions

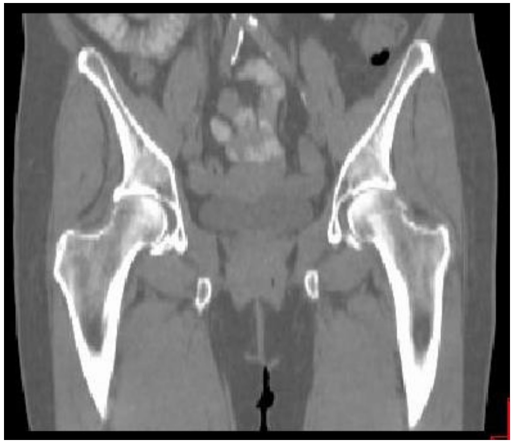
Input image

Function

Output image





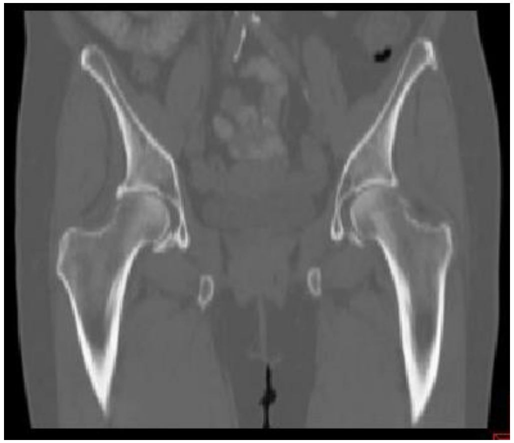


* 2x window in some applications

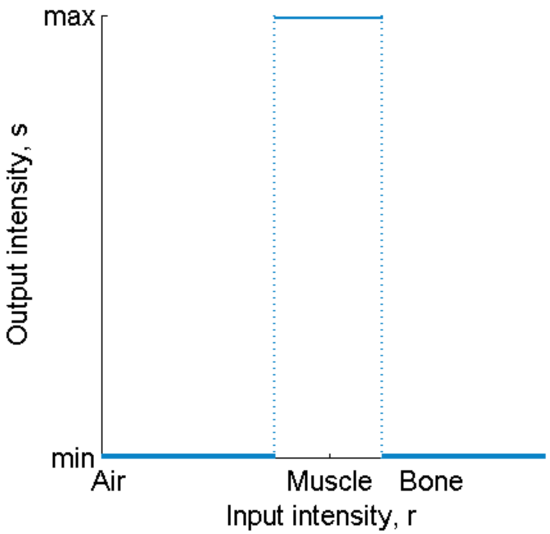
Intensity transformations:
Threshold

- Simplest image segmentation!

Input image



Function



Output image

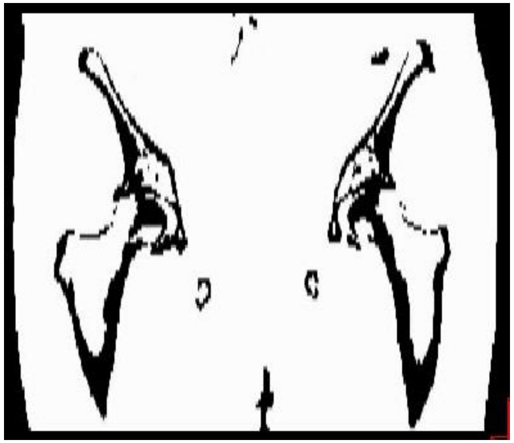
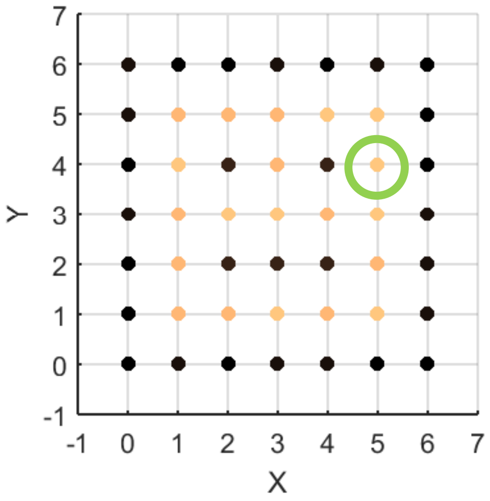


Image as Points + Intensity Value



Besides the intensity value, every pixel has a set of coordinates (x,y)

e.g. (5.0, 4.0)

Spatial transformations are applied to the pixel coordinates!

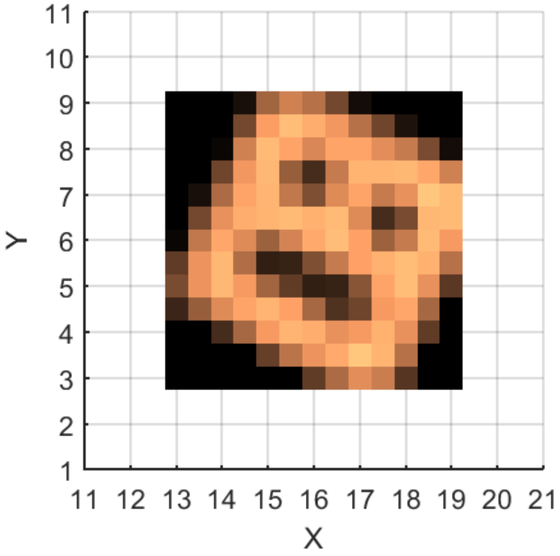
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This is all handled by ndimage.interpolate (and ndimage.rotate)

→ 3D add a new coordinate: z, and T becomes a 4x4 matrix!!

Comparing Interpolators

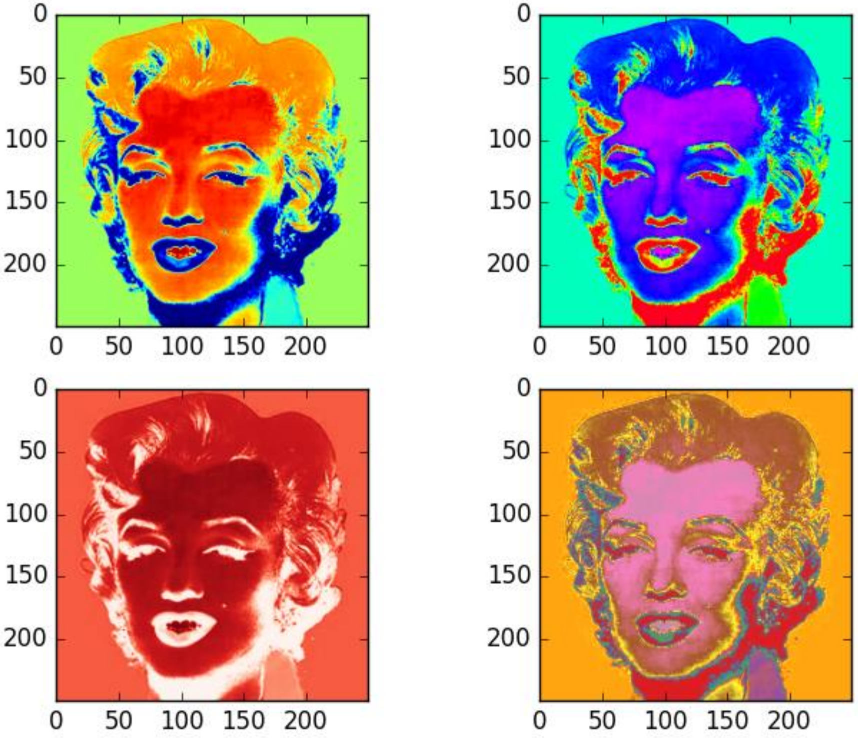
- Nearest neighbor
  - Simplest, fast, less accurate



- Linear
- Polynomial
  - Bit more complicated, slower, ‘more accurate’

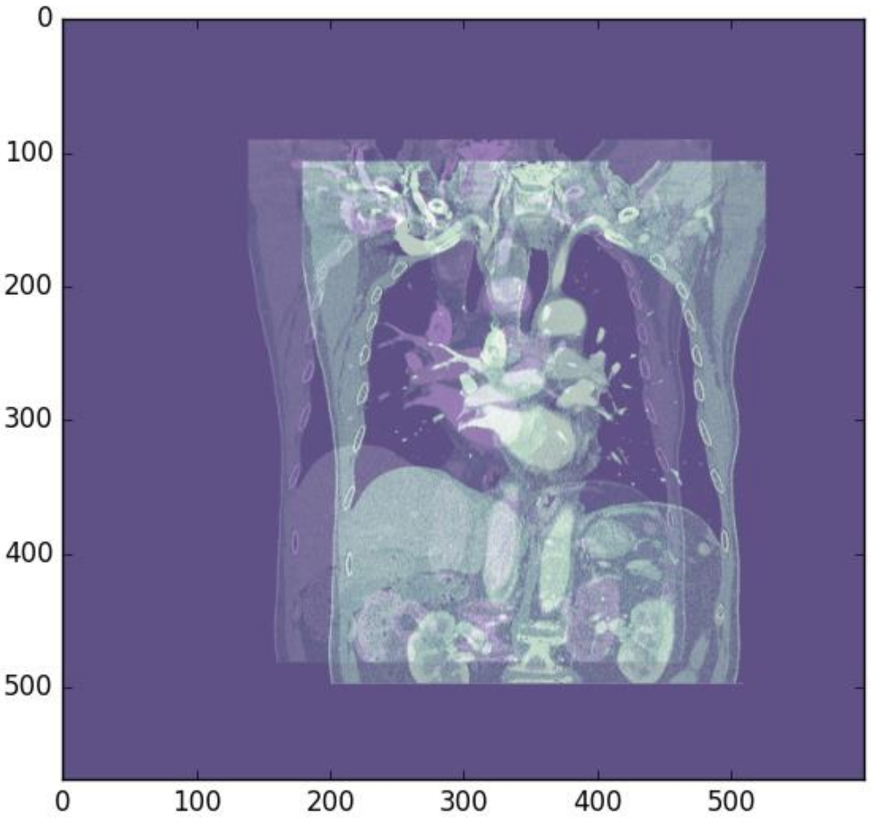
Displaying Images

- We are using imshow from matplotlib
  - Used it a lot yesterday
- Colour maps can be specified with e.g. `cmap=“Greys_r”`
- The alpha option sets the transparency of an image
  - Can also be used in other plots



Displaying Images for fusion

- The most common display for image fusion is green/purple
  - One image uses `cmap="Green_r"`, the other has `cmap="Purple_r"`
- Also have to have some transparency



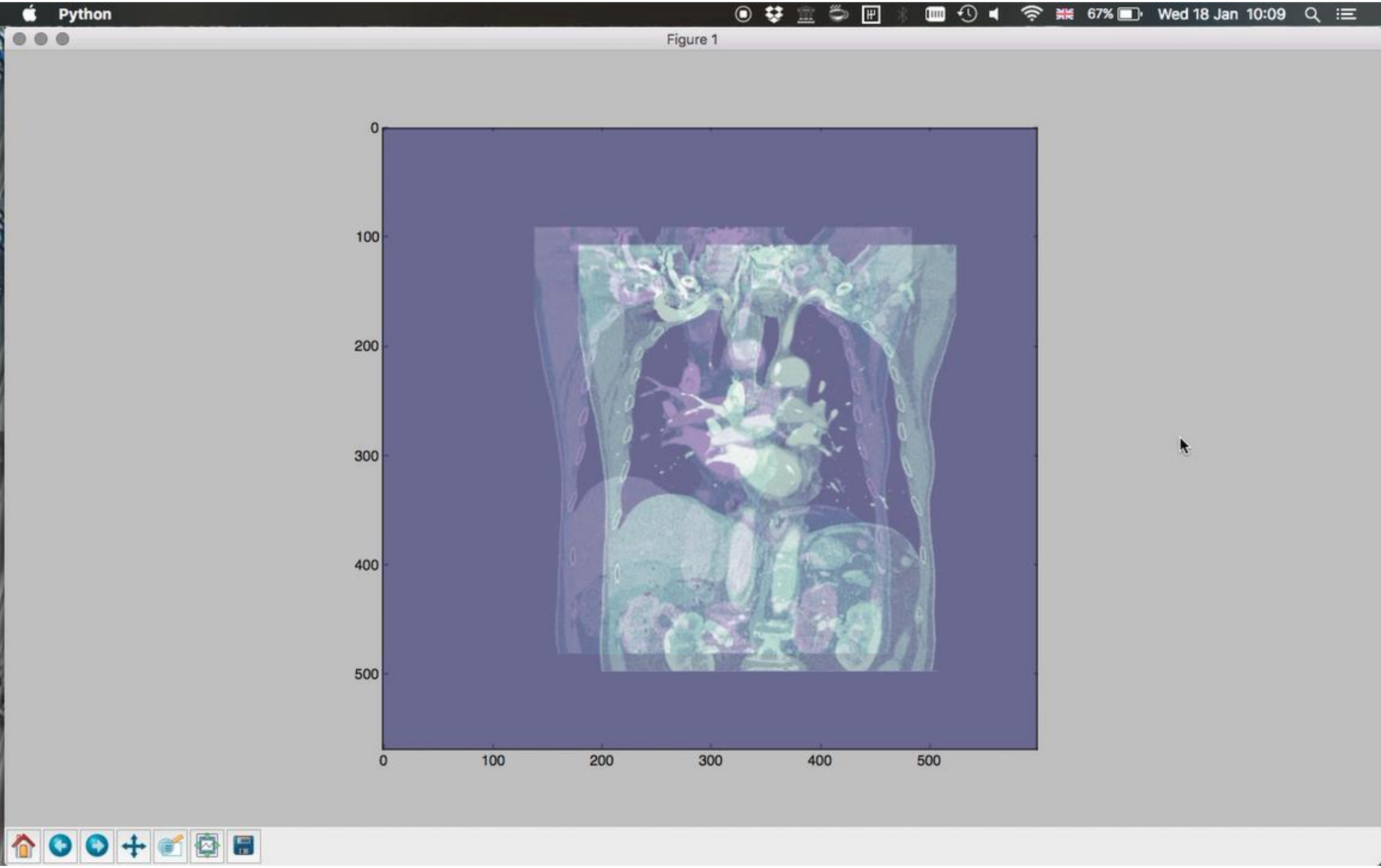
Manual Fusion

- To do our image analysis, we need the images to overlap
  - Line up anatomy on anatomy
- Simplest way is to register the images manually by eye
  - Wrote a fusion code yesterday
  - Works fine for rigid registration
  - What happens if the patient changes shape?
    - We won’t be dealing with this problem!

Manual Fusion in python

- To be able to do manual fusion, we made interactive plots.
- This is done in `matplolib` with ‘event handlers’
    - You did this in the practical yesterday

```
def manualRegister(event):
    if event.key == “up”:
        # Move the image up
    elif event.key == “down”:
       # Move the image down
    …
    elif event.key == “alt+left”:
        # rotate the image?

# Somewhere in the code...
cid = fig.canvas.mpl_connect('key_press_event', manualRegister)
```

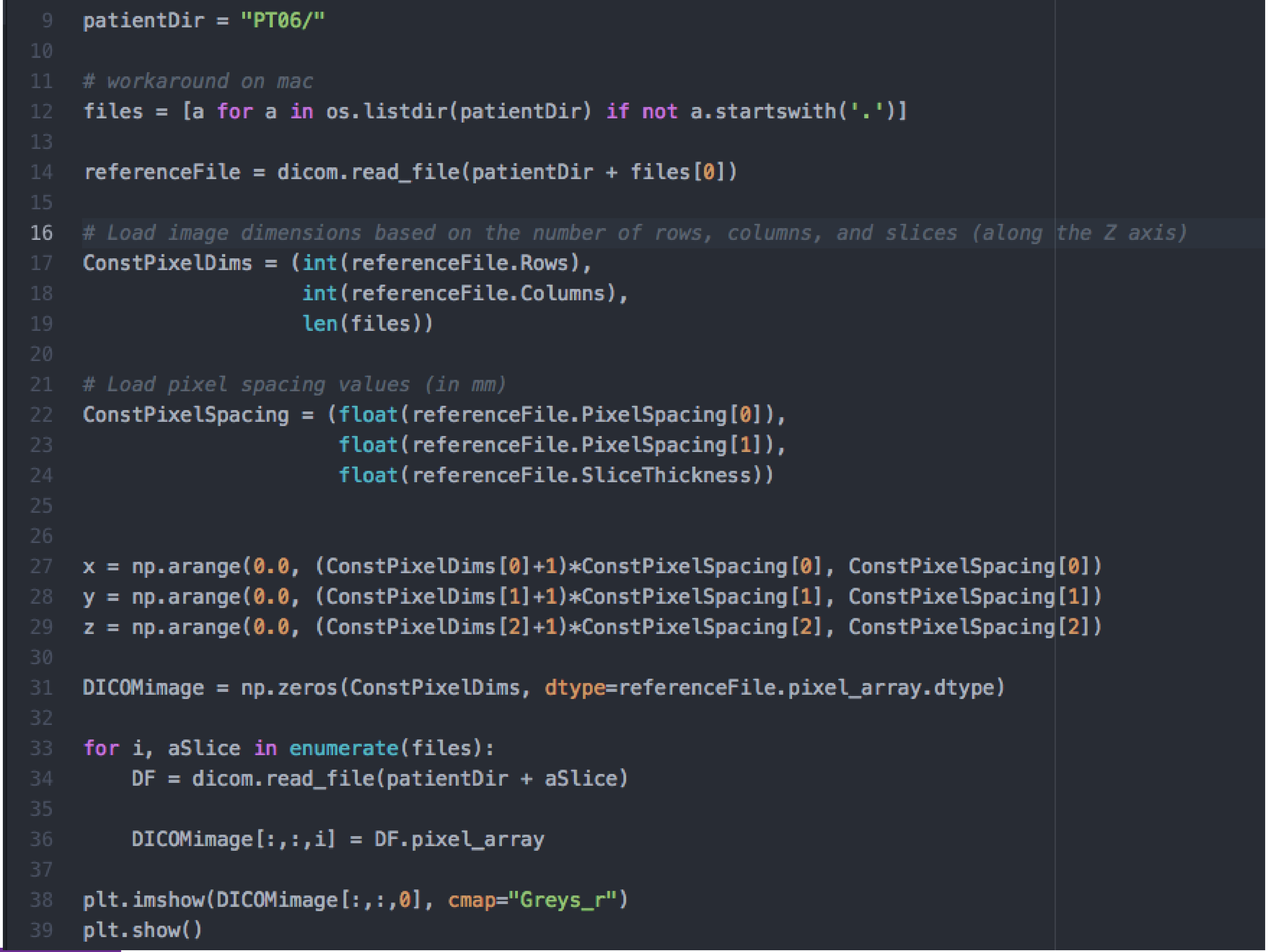

Python
67%
Wed 18 Jan 10:09
Figure 1
0
100
200
300
400
500
0
100
200
300
400
500

Loading DICOM in python

- Recap: DICOM is a file format and communication standard.
- The DICOM specification is very complicated
  - Huge and constantly evolving
  - Optional fields make writing a DICOM reader difficult.
- Python has two libraries:
  - pydicom – main DICOM library
  - pynetdicom – For connecting to DICOM servers

pydicom Example

```
import dicom
import matplotlib.pyplot as plt

patientImage =
dicom.read_file(“419827491.204719.128419.dcm”).pixel_array

plt.imshow(patientImage)
plt.show()
```



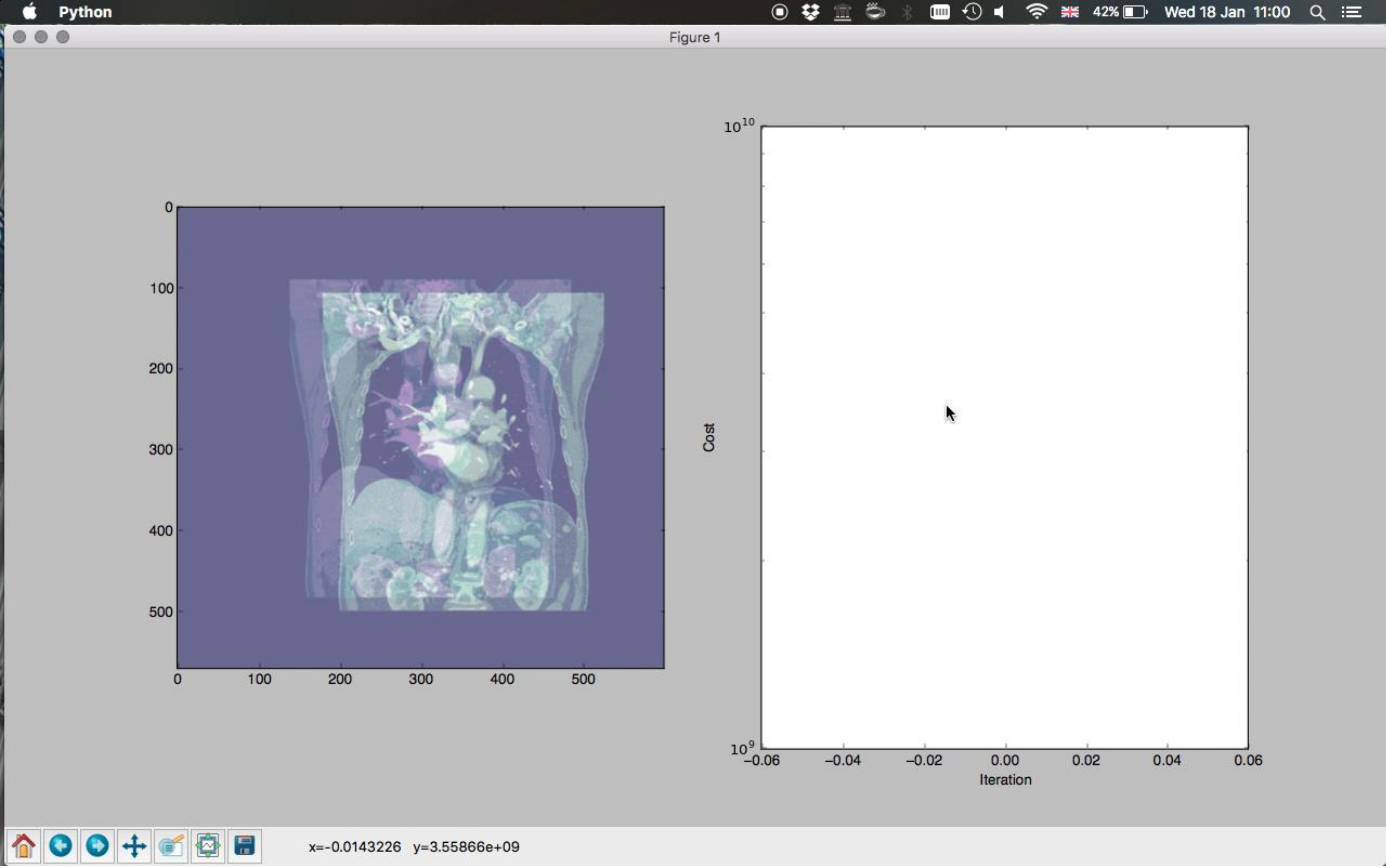
```python
patientDir = "PT06/"

# workaround on mac
files = [a for a in os.listdir(patientDir) if not a.startswith('.')]

referenceFile = dicom.read_file(patientDir + files[0])

# Load image dimensions based on the number of rows, columns, and slices (along the Z axis)
ConstPixelDims = (int(referenceFile.Rows),
                  int(referenceFile.Columns),
                  len(files))

# Load pixel spacing values (in mm)
ConstPixelSpacing = (float(referenceFile.PixelSpacing[0]),
                     float(referenceFile.PixelSpacing[1]),
                     float(referenceFile.SliceThickness))


x = np.arange(0.0, (ConstPixelDims[0]+1)*ConstPixelSpacing[0], ConstPixelSpacing[0])
y = np.arange(0.0, (ConstPixelDims[1]+1)*ConstPixelSpacing[1], ConstPixelSpacing[1])
z = np.arange(0.0, (ConstPixelDims[2]+1)*ConstPixelSpacing[2], ConstPixelSpacing[2])

DICOMimage = np.zeros(ConstPixelDims, dtype=referenceFile.pixel_array.dtype)

for i, aSlice in enumerate(files):
    DF = dicom.read_file(patientDir + aSlice)

    DICOMimage[:,:,i] = DF.pixel_array

plt.imshow(DICOMimage[:,:,0], cmap="Greys_r")
plt.show()
```

pynetdicom

- Allows you to connect to a DICOM server and pull images from it.
  - Would have been really cool for the practical
- pynetdicom is quite tricky to set up!
  - Work with local files instead
- If we have time/anybody is interested, we can try it later

Cost Functions

- A cost function is just a function that tells you how bad the match is
  - Also known as a metric
- Look at the two images, how much do they overlap?
- How might you measure it?
  - Sum of square differences (simple and effective!)
  - Cross entropy
  - Mutual information



x=-0.0143226 y=3.55866e+09

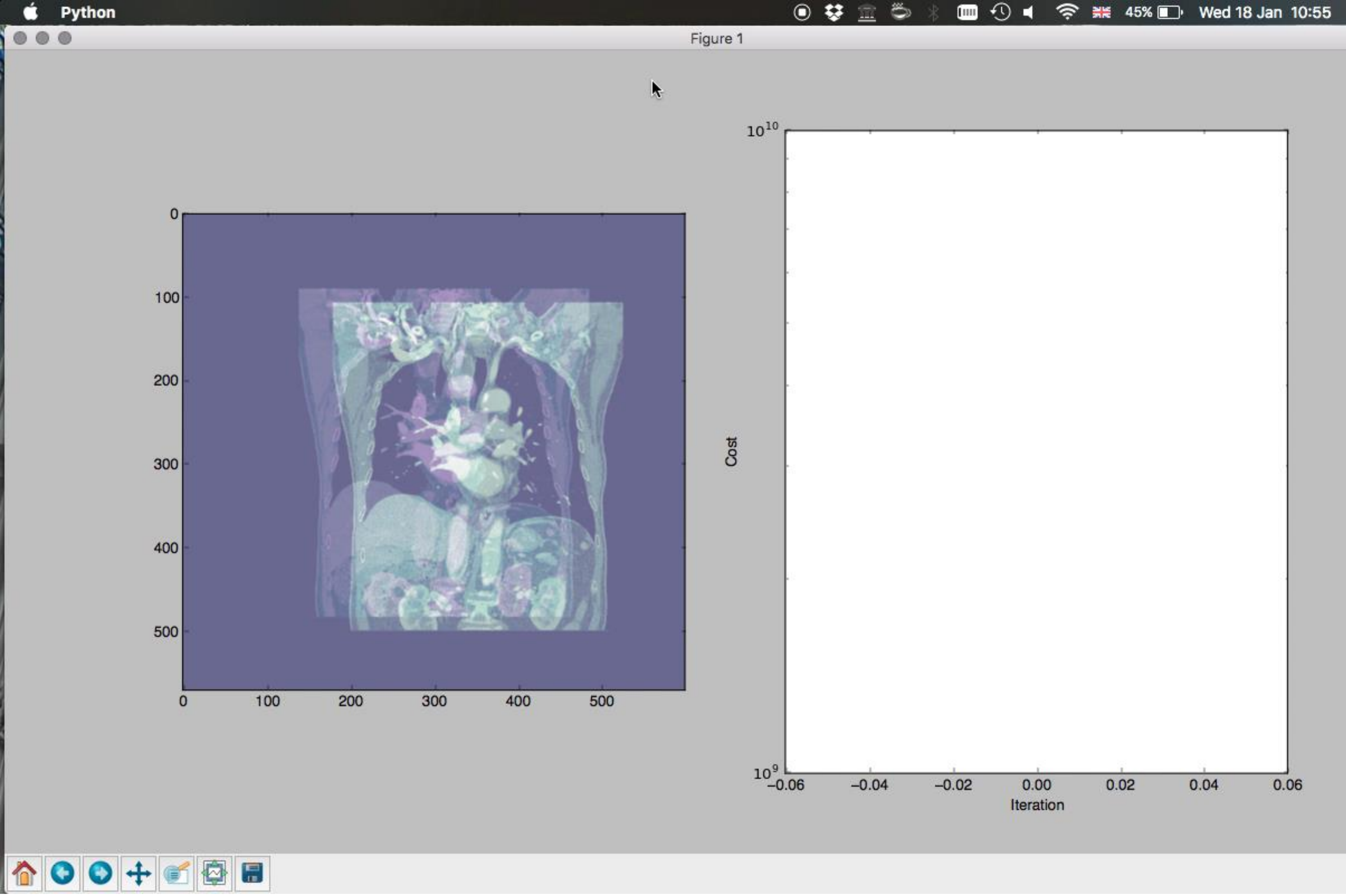
Cost Functions

```
def costFunction(image1, image2):
    “””
    Returns a cost function based on the two images
    “””
    return np.mean((image1 – image2)**2)
```

What is this cost function called?

Automatic Fusion

- Manual fusion is useless for large datasets
  - Need to register thousands of images
  - Not practical manually
- Use cost function to enable automatic registration
  - Need to check the accuracy of your registration somehow



The University of Manchester

The Christie NHS Foundation Trust

Automatic Fusion

- Automatic fusion is just an optimisation problem
  - Minimising the cost function
- Optimisation is a tricky subject…
- Fortunately python has a library for this!
  - `scipy.optimize`
- `scipy.optimize` contains many algorithms
  - We will be testing a few of them later

Automatic Fusion – python skeleton

```
def shiftImages(shifts, opt. arguments to update plot ):
      # Interpolate & rotate image
      # test if opt. args are present, update plot
      # if they are

# Do a brute force optimisation (for example)
res = brute(shiftImages ((xLow, xHi),(yLow, yHi)))

# do a final plot update call to shift the images
# Note – it’s a different function!
shiftImages2(res, < opt. args. >)
```

Automatic Fusion – python in action

Automatic fusion

- Yesterday we made the manual fusion code
  - Wrote two functions: `eventHandler` and `shiftImages`
- We should be able to repurpose that code now to use in automatic fusion.
  - That’s why we wrote it like that!
  - However, a couple of changes will be needed…

Automatic Fusion

Manual fusion

- Take individual small steps
- A step is in x or y, not both
- Update the global image after each step
- Update a plot after every step

Automatic fusion

- Move image in large shifts
- Shift is a 2D vector (i.e. x and y)
- Need to start from the same location every time
- Don’t usually update a plot

Cropping an image

- We saw yesterday how to crop parts of an image
  - Use numpy array slicing, eg `image[100:200, 100:200]`
- Today we will use an interactive method to get the indices, and crop a region of image
  - Code is already written!
  - You just need to link up the event handlers to the right event

What we haven’t mentioned

- So far we have done rigid registration
  - Assumes the patient is exactly the same shape in each image
- To do proper image analysis, we need to use deformable registration
  - Allows us to handle weight loss and other changes
- Implementation is a bit beyond this course!
  - Use a library like SimpleElastix if you need it

This afternoon’s practical

- Hopefully, as we treat a patient, the tumour shrinks
- Part of the treatment is frequent imaging, so we should be able to observe the tumour shrinkage
- Your task:
    - Write an automatic image fusion code
    - Use it to fuse a set of images
    - Extract tumour shrinkage information

This afternoon’s practical

- You will need:
  - DICOM image loading
  - Image display in green/purple
  - Manual image fusion
  - Cost function
  - Automatic image fusion
  - Cropping
  - Thresholding

Extension!

- Last year, we developed this:
  - https://github.com/afg1/SoundCost
  - It’s a manual image registration system
  - Cost function is used to control a beep
  - Gets faster as cost gets lower
    - Alien Image Registration Engine
- Try to think of another pointless modification to your image registration tool
  - Example: frequency modulated with cost value?

More resources

- An ebook about python: https://automatetheboringstuff.com/
- StackOverflow – the answer to pretty much any question is here: http://stackoverflow.com/questions/tagged/python
- StackOverflow Documentation – a Wikipedia-like documentation of python (and loads of other languages) http://stackoverflow.com/documentation/python/topics
- List of cool python libraries/scripts: https://github.com/vinta/awesome-python
- Numpy documentation: https://docs.scipy.org/doc/numpy/

See you in the practical!