Usage of SimpleITK

SIMPLEITK

- A simplified version of ITK (Insight Tool Kit) for scientific image processing, segmentation, and registration

- Ideal for medical imaging: X-rays, CT, PET, MRI, and US

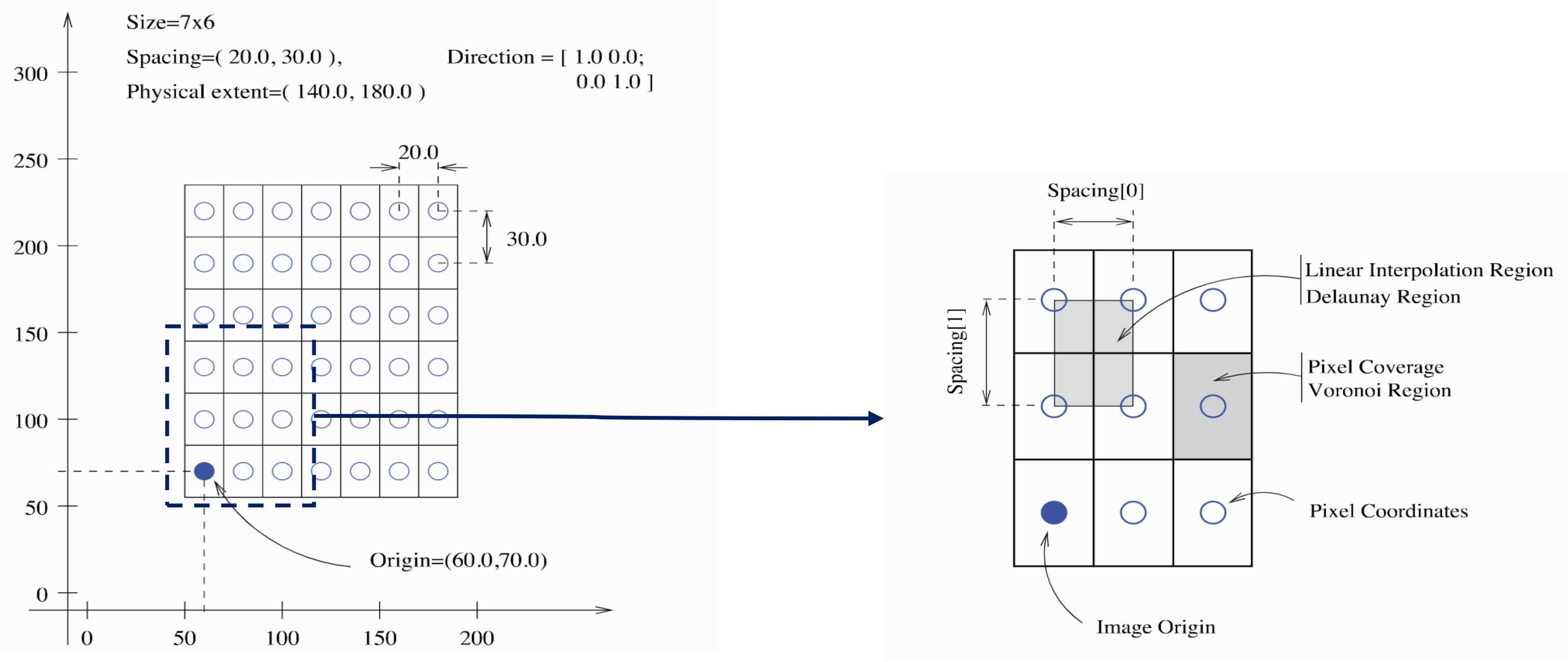
- Available in Python:

  pip install SimpleITK | conda install simpleitk

Image fundamentals

- An image in SITK ≠ than in cv2 or scikit-image
  - SITK = set of points on a grid occupying a physical region in the space
  - Others = array

- SITK takes into account:
  1. Pixel/voxel spacing information
  2. Location in the physical space



- Image spacing:

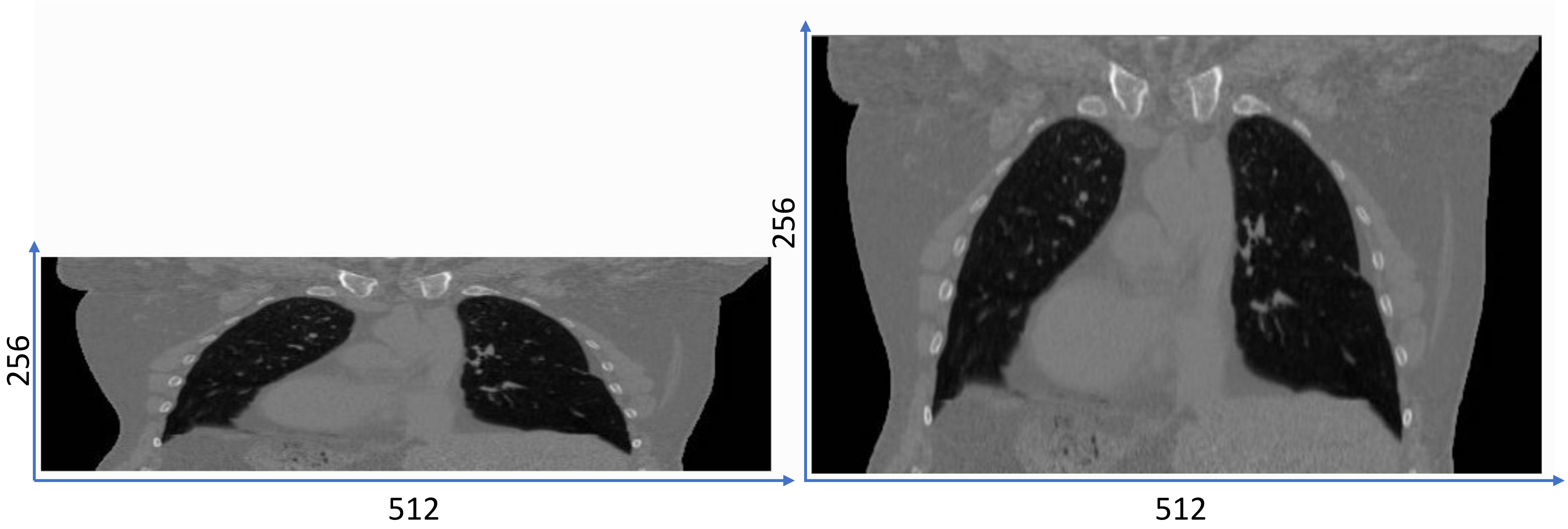
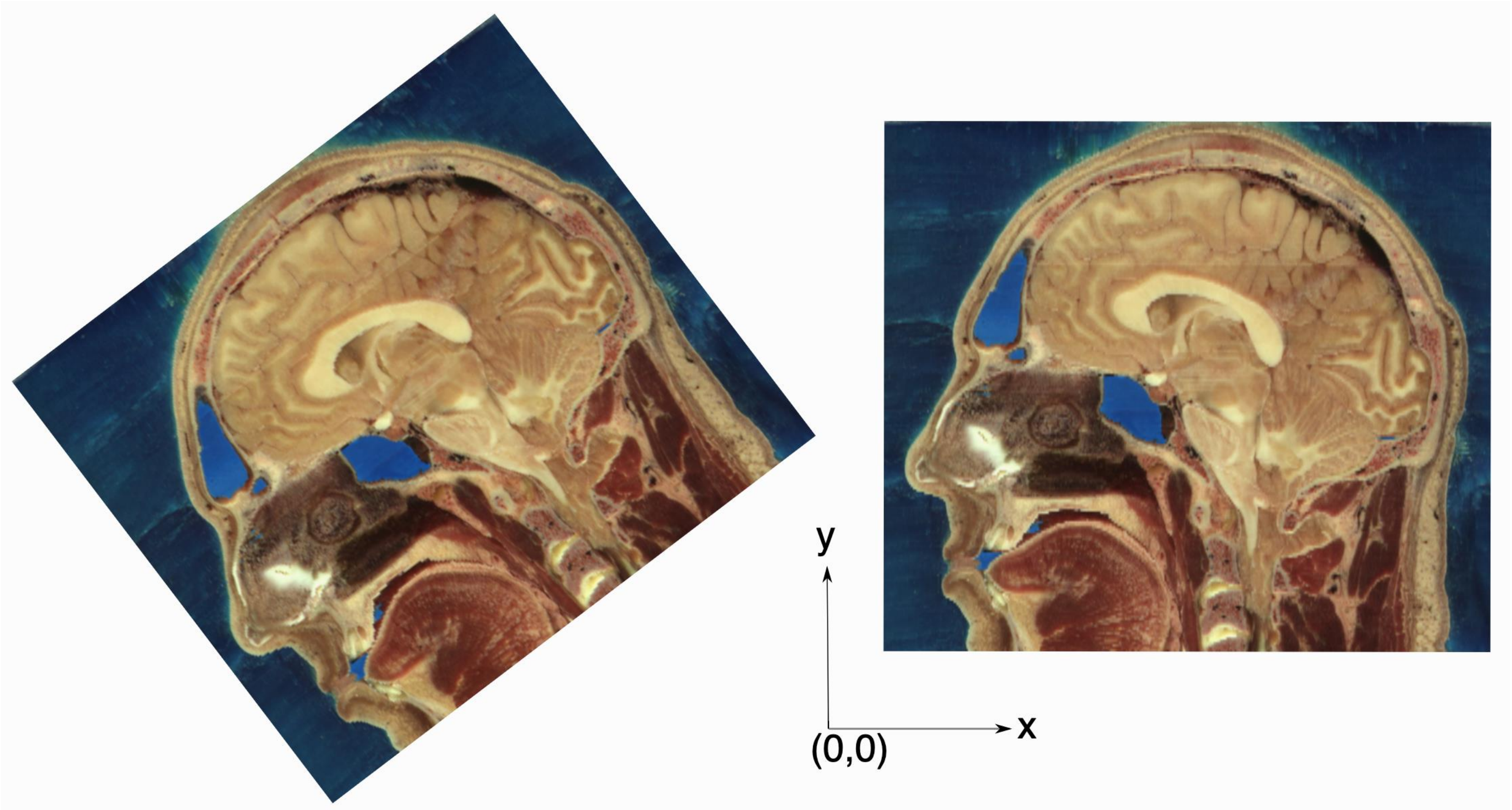


Image spacing = (1.0, 1.0)

Image spacing = (0.97, 2.0)

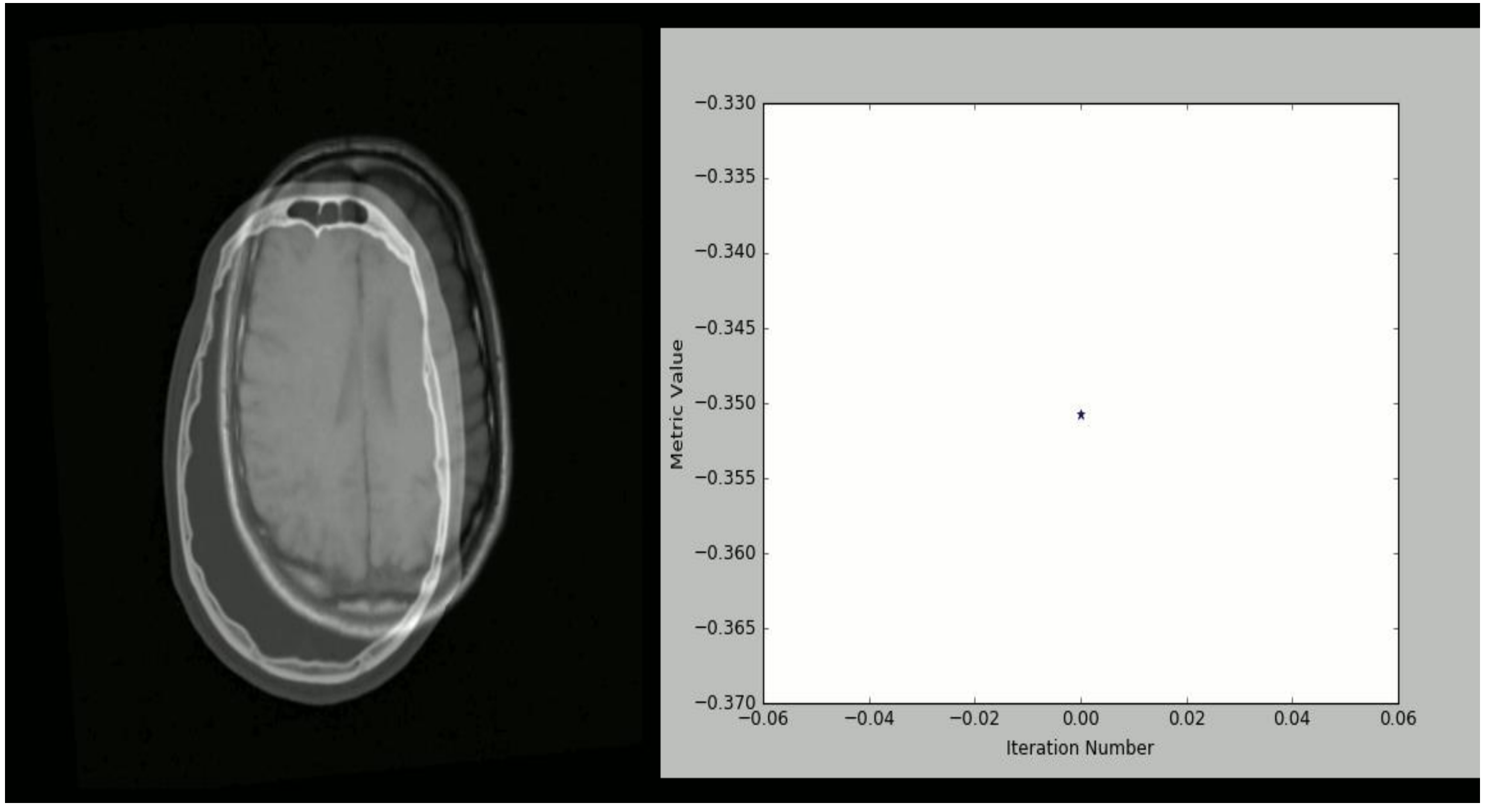
- Image origin and direction:



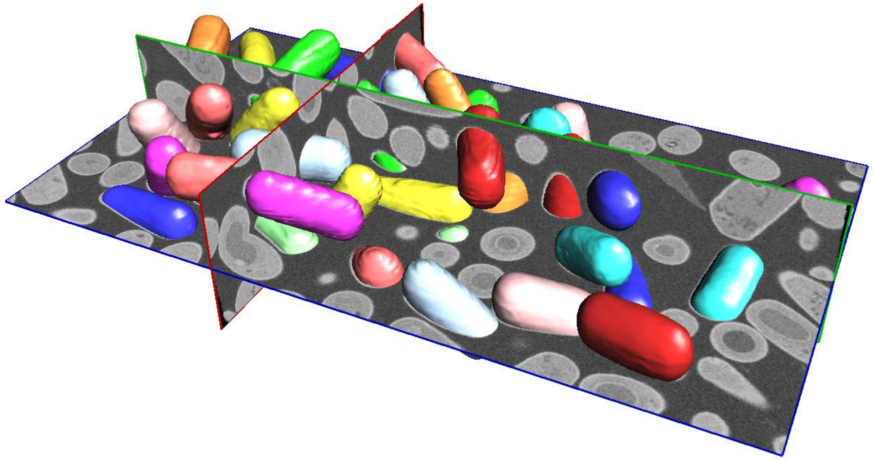
- Image origin = (-136.3, -20.5)
- Orientation matrix = (0.7, -0.7, 0.7, -0.7)

- Image origin = (16.9, 21.4)
- Orientation matrix = (1, 0, 0, 1)

- Image registration:



- Image segmentation:



How to read an image?

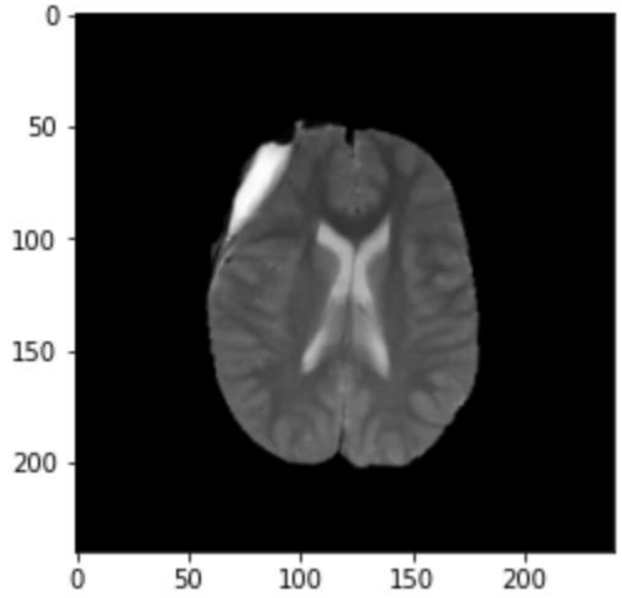
1. import SimpleITK as sitk
2. image = sitk.ReadImage(path_to_image)

Conversion between SimpleITK and NumPy

From SimpleITK to NumPy:

1. image = sitk.ReadImage(path_to_image)
2. array = sitk.GetArrayFromImage(image)
   - array.shape -> (155, 240, 240)

```
plt.imshow(array[87, :, :], cmap='gray')
plt.show()
```



Reading an image

- Multiple image formats:
  - Standard ones: .jpeg, .png, .tiff, etc
  - Medical ones: .mha, .nii, .dcm

- image = sitk.ReadImage(path_to_image)
  - Default = '16-bit signed integer'

- image = sitk.ReadImage(path_to_image, sitk.sitkUInt8)

sitkUInt8	Unsigned 8 bit integer
sitkInt8	Signed 8 bit integer
sitkUInt16	Unsigned 16 bit integer
sitkInt16	Signed 16 bit integer
sitkUInt32	Unsigned 32 bit integer
sitkInt32	Signed 32 bit integer
sitkUInt64	Unsigned 64 bit integer
sitkInt64	Signed 64 bit integer
sitkFloat32	32 bit float
sitkFloat64	64 bit float

Practical Session 3

Image enhancement and denoising

Exercise 1: Image histogram

1. Linear stretching *

2. Histogram equalization *

3. CLAHE (skimage implementation)

Exercise 1: Image histogram

1. Linear stretching *

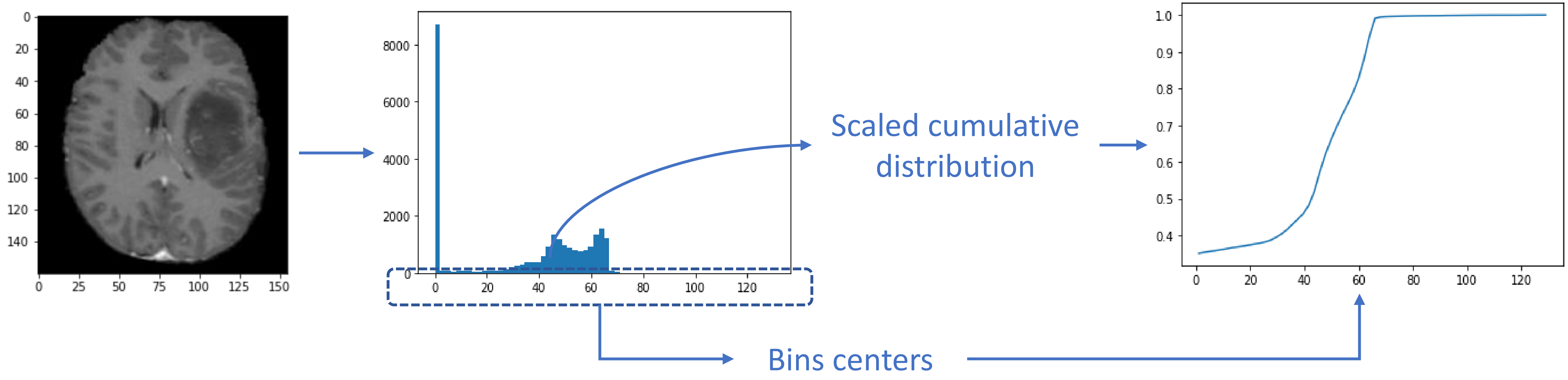
$$j = \frac{i - i_{\min}}{i_{\max} - i_{\min}}(j_{\max} - j_{\min}) + j_{\min}$$

$$j_{min} = 0 \quad j_{max} = 1$$

$$i_{min} = P_5 \quad i_{max} = P_{95}$$

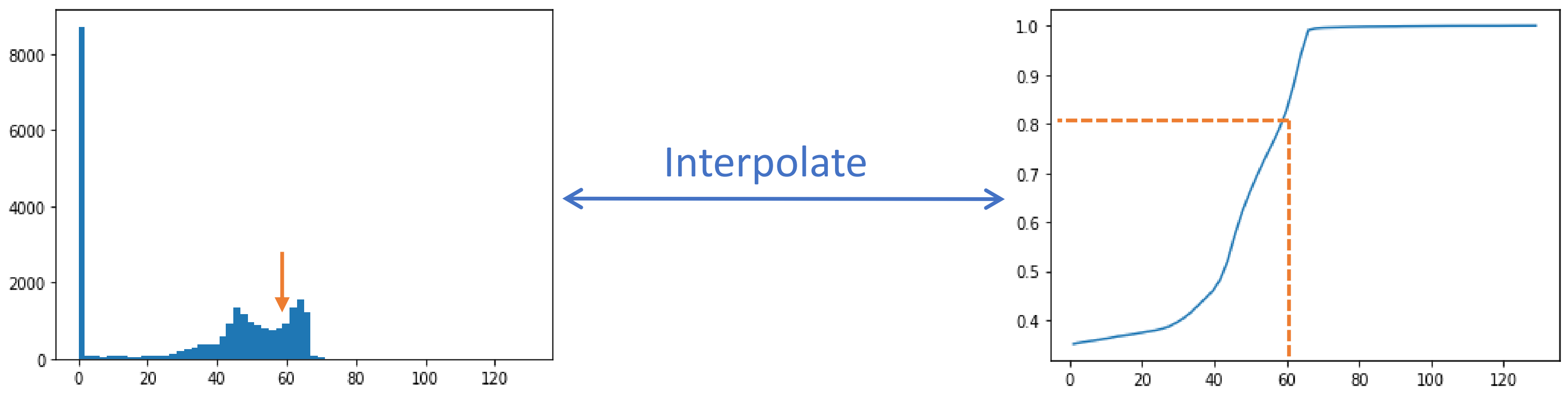
Exercise 1: Image histogram

2. Histogram equalization*

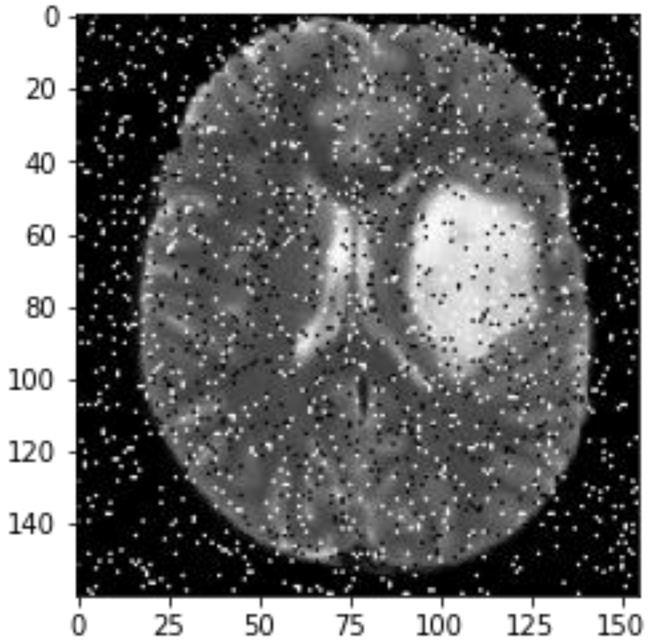


Exercise 1: Image histogram

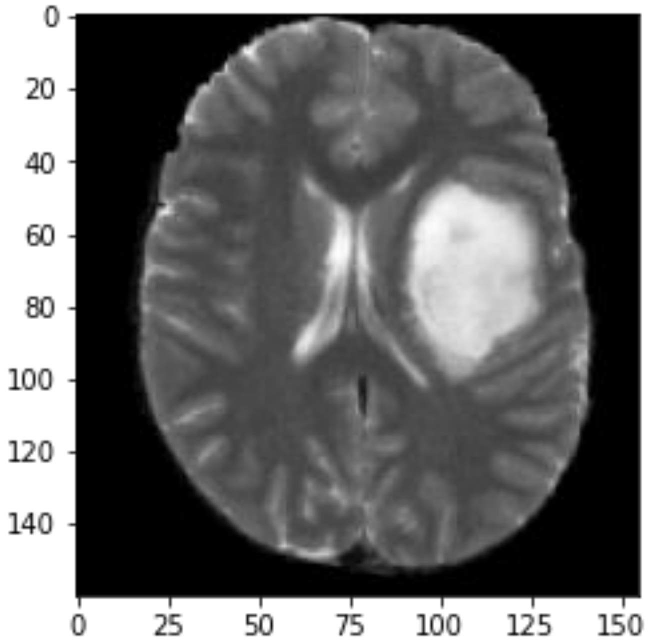
2. Histogram equalization*



Exercise 2: Image denoising



1. Gaussian filter
2. Median filter
3. Mean filter *

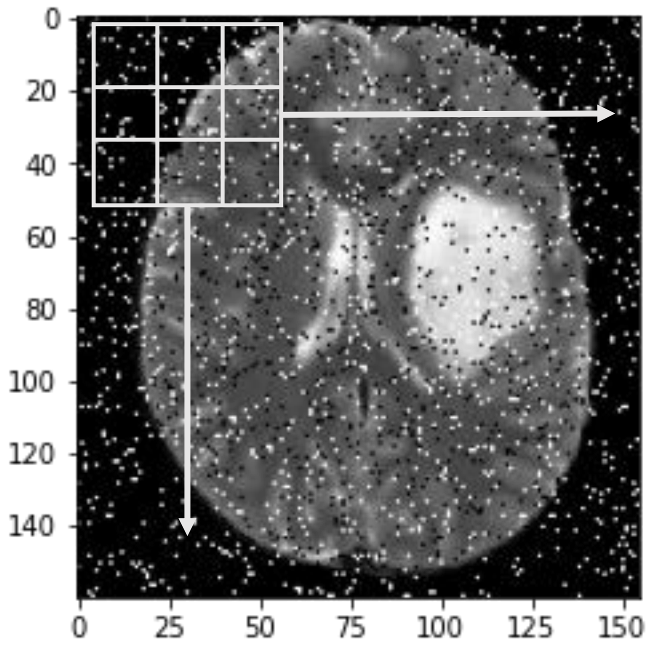


Exercise 2: Image denoising

3. Mean filter *

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Convolve



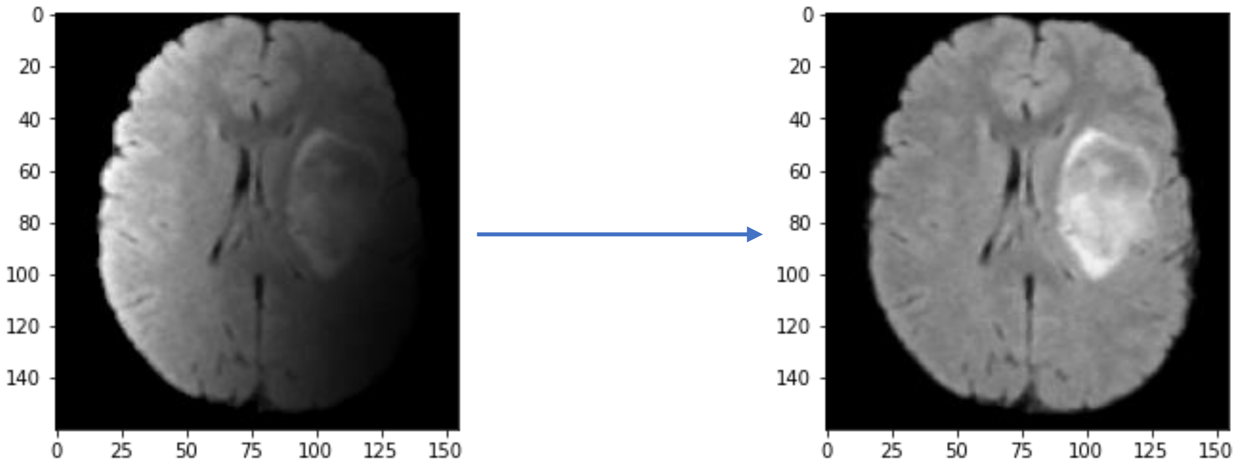
Exercise 3: HUM algorithm

- Remove bias noise

$$f_{i,j} = g_{i,j} \cdot \frac{\mu}{\mu_{i,j}},$$

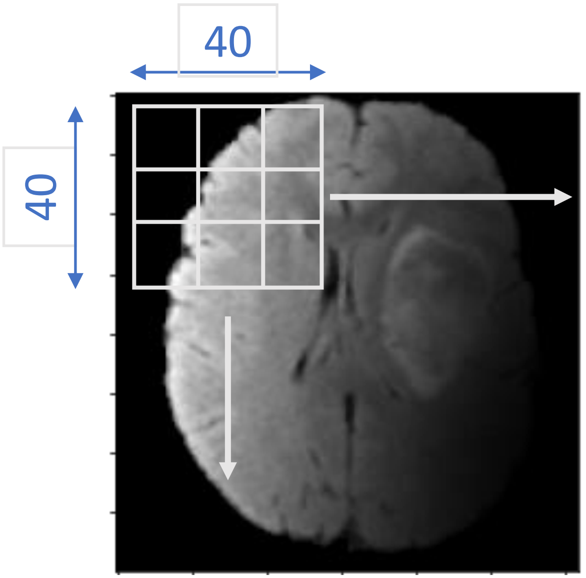
$f_{i,j} = output\ pixel\ intensity$ $\quad \mu = global\ mean\ intensity$

$g_{i,j} = input\ pixel\ intensity$ $\quad \mu_{i,j} = local\ mean\ intensity$



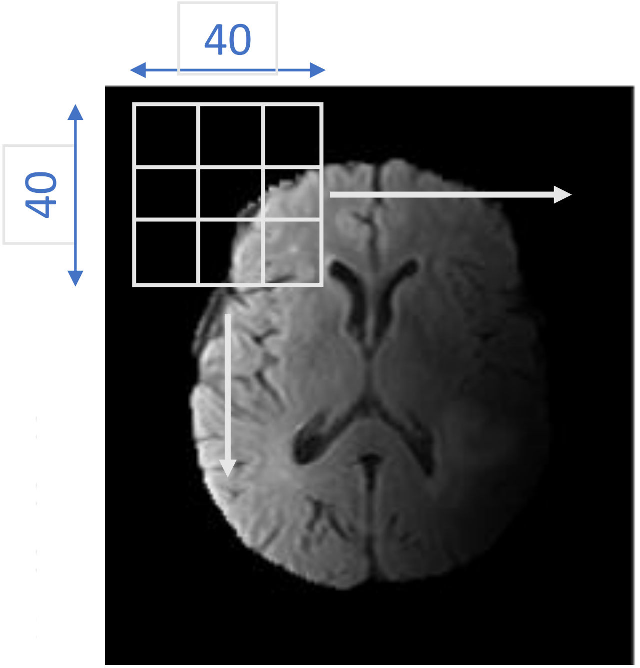
Exercise 3: HUM algorithm

1. Direct implementation:
    i. Get mean intensity (global)
    ii. Define a window of 40 x 40
    iii. Get mean intensity (local)
    iv. Compute HUM
    v. Move the window, and repeat



Exercise 3: HUM algorithm

2. Pad image:
    i. Pad the image
    ii. Get mean intensity (global)
    iii. Define a window of 40 x 40
    iv. Get mean intensity (local)
    v. Compute HUM
    vi. Move the window, and repeat



Exercise 3: HUM algorithm

3. Pad + threshold:
    i. Pad the image
    ii. Filter pixels below a threshold (10)
    iii. Get mean intensity (global)
    iv. Define a window of 40 x 40
    v. Get mean intensity (local)
    vi. Compute HUM
    vii. Move the window, and repeat