ALS User Meeting 2022

This notebook describes methods to extract key information from microCT image stacks

- Histograms
- Alternative interactive tools and hist
- Maximum intensity projection
- Largest connected component

Created by Dani Ushizima, CAMERA, LBNL - Aug 1st 2022

```
%matplotlib inline
```

```
import numpy as np
from scipy import ndimage as ndi
import fnmatch,os
import matplotlib.pyplot as plt
from glob import glob

from skimage import img_as_ubyte, filters, morphology, exposure, io
from skimage.filters import threshold_isodata
from skimage.transform import pyramid_expand
from skimage.measure import regionprops,label
```

1. Read a microct image

- from url
- from NERSC
- from Google drive

Read from NERSC

- discard this portion if running in Colab

```
datapath = "/global/cfs/cdirs/als/users/yourname/yourdata/" #update these values
!ls -lt "$datapath"
```

```
image = io.imread(datapath+'bead_pack.tif')
```

Read from Google drive

- discard this portion if running at NERSC

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
datapath = "/content/drive/My Drive/Colab Notebooks/ALS User Meeting 2022 colab/data/'
!ls -lt "$datapath"
```

```
total 7842
-rw------- 1 root root 8025493 Aug 11 16:46 bead_pack.tif
drwx------ 2 root root    4096 Aug 11 16:35 concrete
```

```
def loadFileNames(path,extension):
  ''' Return filename after using colab files.upload - work for 1 file'''
  fnames = glob(path+extension)
  fnames.sort()
  print(path);
  print(f"Number of files: {len(fnames)}")
  return fnames
```

```
extension = '*tif'
files = loadFileNames(datapath+'concrete/',extension)
```

```
/content/drive/My Drive/Colab Notebooks/ALS User Meeting 2022 colab/data/concret
Number of files: 20
```

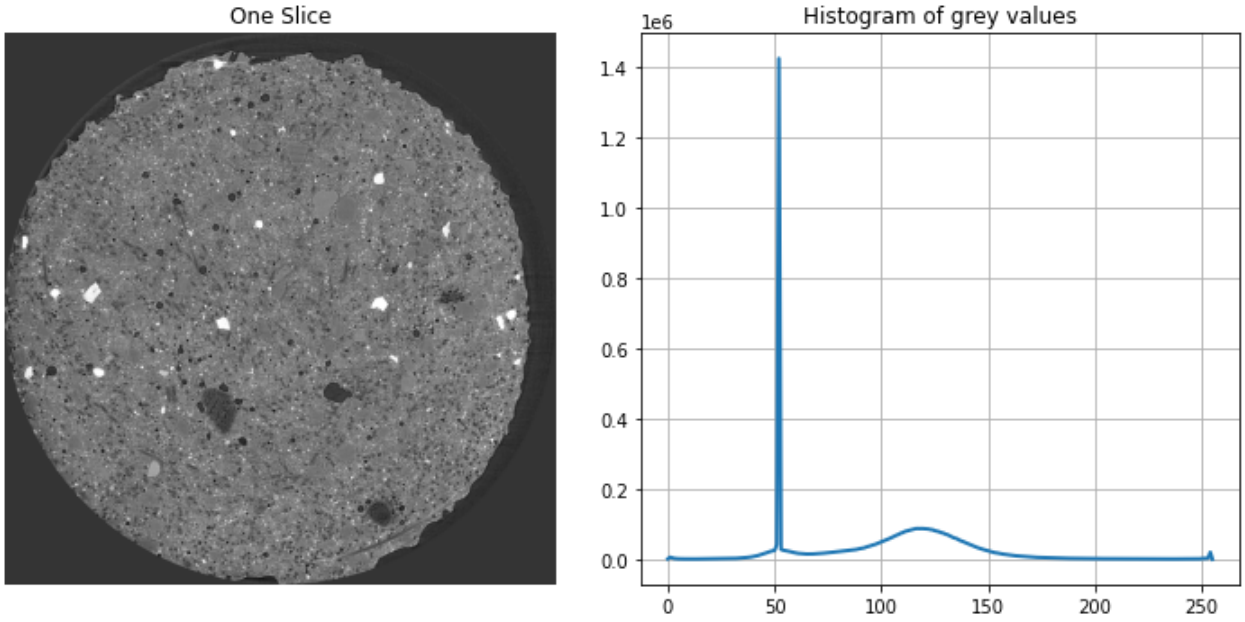
2. Image histograms

- histogram of a single slice
- histogram per slice for stack
- histogram of full stack

```
def seeHistOneSlice(aSlice):
        '''See the histogram of a particular slice after its masking, followed by enha
        hist, hist_centers = exposure.histogram(aSlice)
        fig, ax = plt.subplots(ncols=2, figsize=(10, 5))

        ax[0].imshow(aSlice, cmap=plt.cm.gray, vmin=np.min(aSlice), vmax=np.max(aSlice
        ax[0].set_title('One Slice')
```

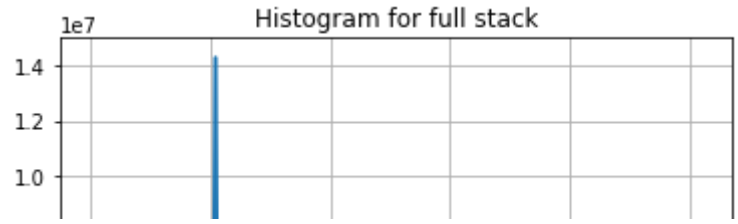
```
        ax[0].axis('off')

        ax[1].plot(hist_centers, hist, lw=2)
        ax[1].set_title('Histogram of grey values')
        ax[1].grid()

        plt.tight_layout()
```

```
n = 5
aslice = io.imread(files[n])
seeHistOneSlice(aslice)
```



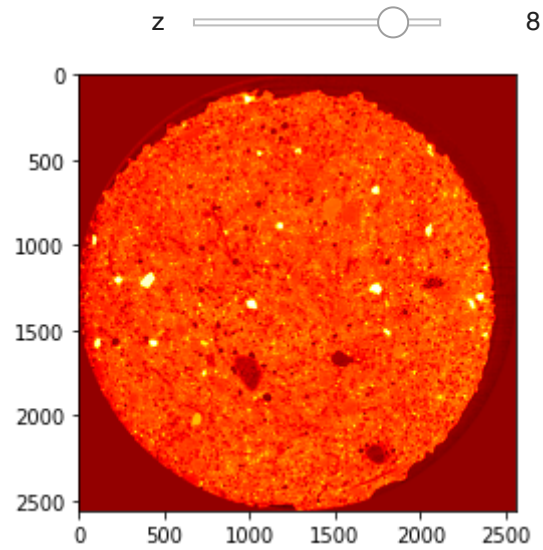

```
#Load full stack - if too large, you can modify functions below to read one slice at a
ic = io.ImageCollection(files,conserve_memory=True)
fullstack = ic[0:10].concatenate()
```

```
hist, hist_centers = exposure.histogram(fullstack)
plt.plot(hist_centers, hist, lw=2)
plt.title('Histogram for full stack')
plt.grid()
```



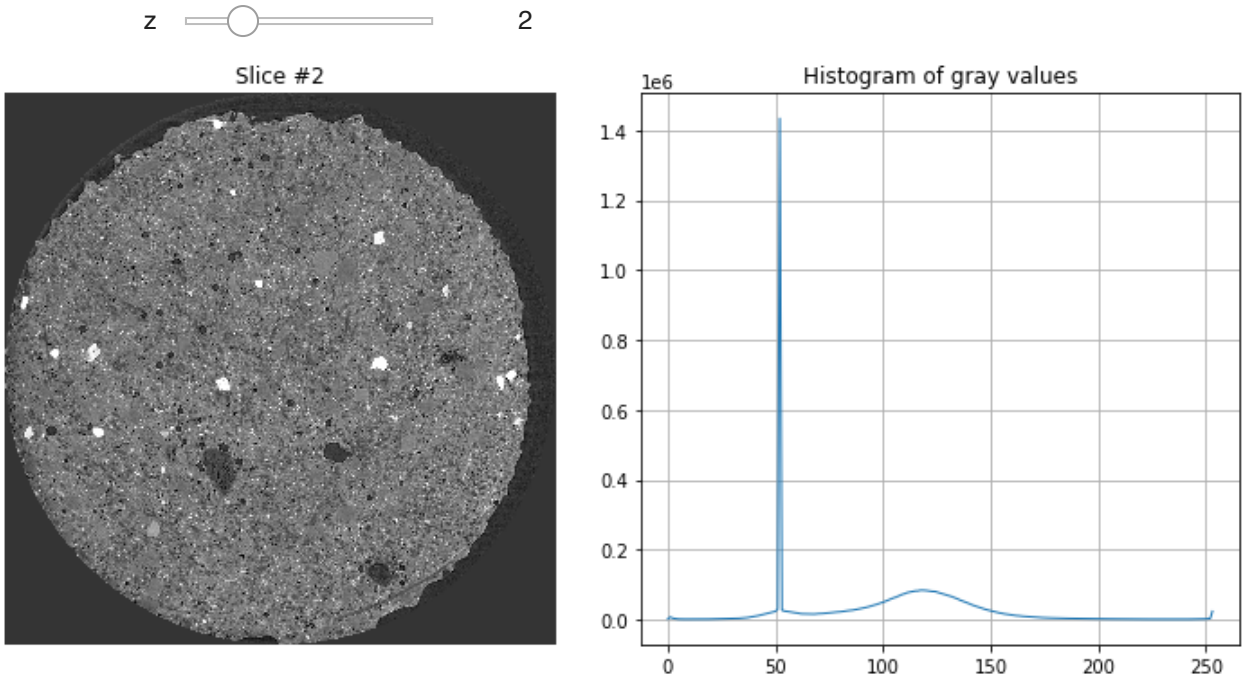
What if visualizing hist per slice?

```
#Scroll through stack as with plotly, usually faster this way
from ipywidgets import interact,IntSlider

def slicer(z):
    plt.imshow(fullstack[z,:,:], cmap='hot')


maxz,_,_ = fullstack.shape
interact(slicer, z=IntSlider(min=0, max=maxz, step=1, value=maxz//2));
```




```
#Scroll through stack and visualize histogram
def seeHistSlice(img,nslice):
    subimage = img[nslice,:,:]
    hist = np.histogram(subimage, bins=np.arange(subimage.min(), subimage.max()))

    fig, ax = plt.subplots(ncols=2, figsize=(10, 5))

    ax[0].imshow(subimage, interpolation='nearest', cmap=plt.cm.gray)
    ax[0].axis('off')
    ax[0].set_title('Slice #'+str(nslice))

    ax[1].plot(hist[1][:-1], hist[0], lw=1)
    ax[1].set_title('Histogram of gray values')
    ax[1].grid()
```

```
    plt.tight_layout()

def slicingHist(b):
    def slicer(z):
        seeHistSlice(b,z)
    interact(slicer, z=IntSlider(min=0,max=len(b)-1,step=1,value=len(b)//2));


slicingHist(fullstack)
```

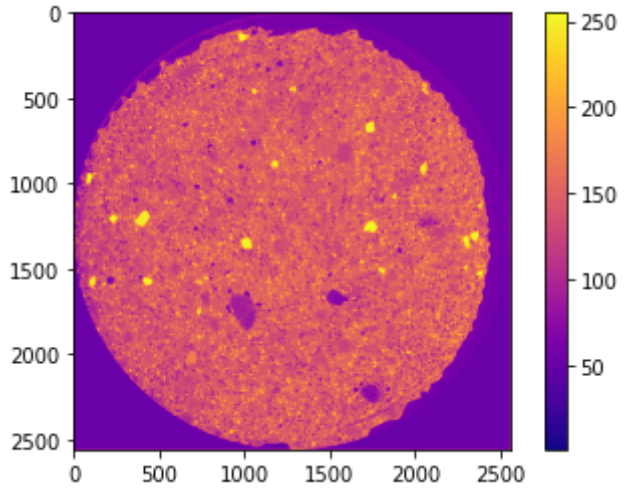


3. Maximum Intensity Projection (MIP)

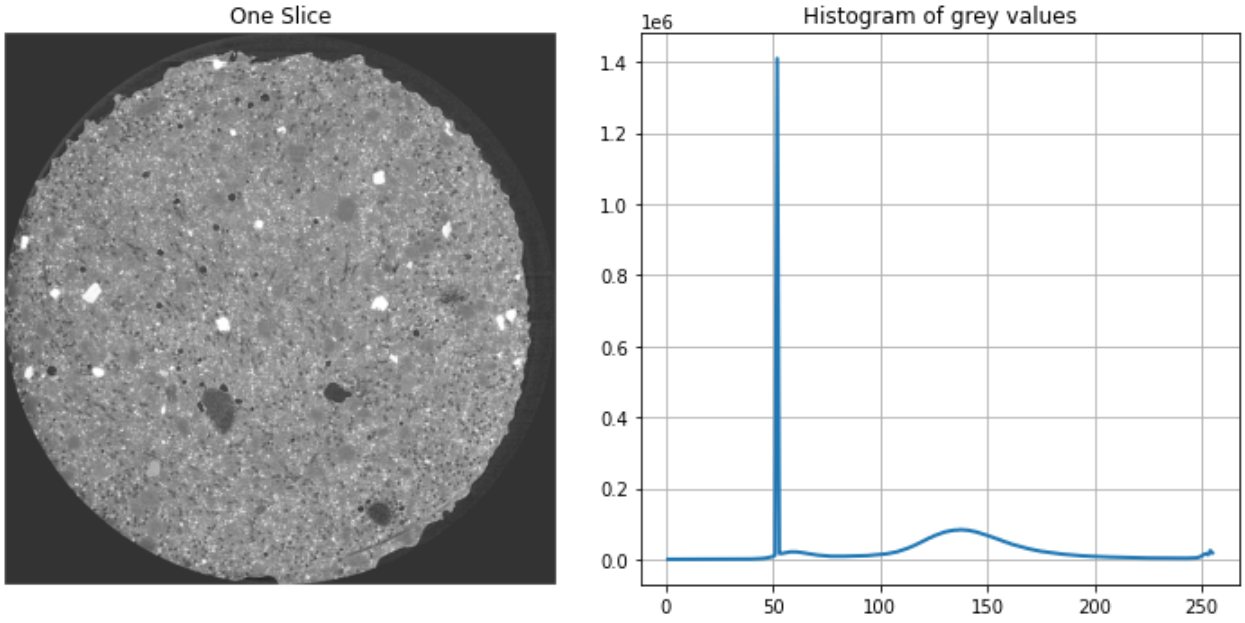
link paper

```
def maxProjection(stack,ndown):
        #max projection aka. z-max
        IM_MAX = stack[0,::ndown,::ndown]
        nslices,_,_ = stack.shape
        for n in np.arange(1,nslices,ndown): #ndown here for low variation on Z
            IM_MAX = np.maximum(IM_MAX, (stack[n,::ndown,::ndown]))
        return IM_MAX


maxProjectionfullstack = maxProjection(fullstack,ndown=1)
plt.imshow(maxProjectionfullstack,cmap='plasma')
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7f9036076650>
```

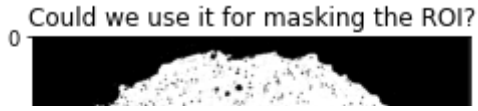



```
seeHistOneSlice(maxProjectionfullstack)
```




```
plt.imshow(maxProjectionfullstack>100,cmap='gray')
plt.title('Could we use it for masking the ROI?')
```

```
Text(0.5, 1.0, 'Could we use it for masking the ROI?')
```

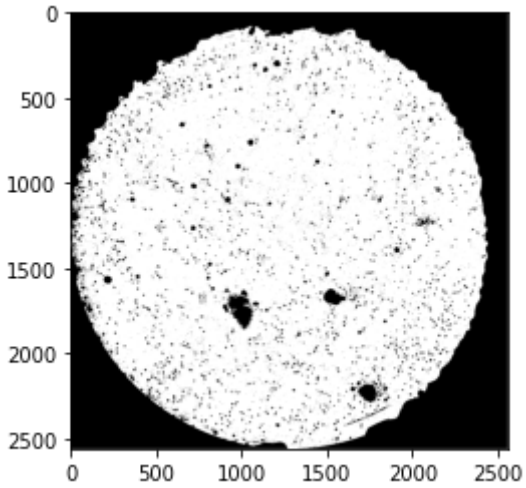


Determine largest connected component

- if slices have common center, MIP can help determine ROI
- calculating a mask
- masking a slice

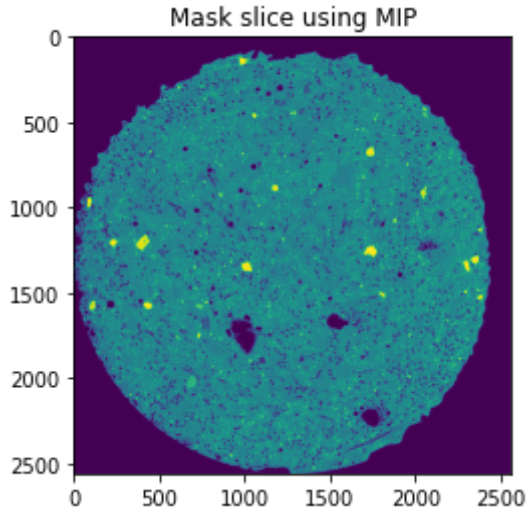
```
def getLargestCC(segments):
        '''Return the largest connected component from image'''
        labels = label(segments)
        largestCC = labels == np.argmax(np.bincount(labels.flat, weights=segments.flat
        return largestCC


lcc = getLargestCC(maxProjectionfullstack>100)


plt.imshow(lcc,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f9035bb5790>
```




```
plt.imshow(aslice*lcc)
plt.title('Mask slice using MIP')
```

```
Text(0.5, 1.0, 'Mask slice using MIP')
```



✓ 1s completed at 2:51 PM