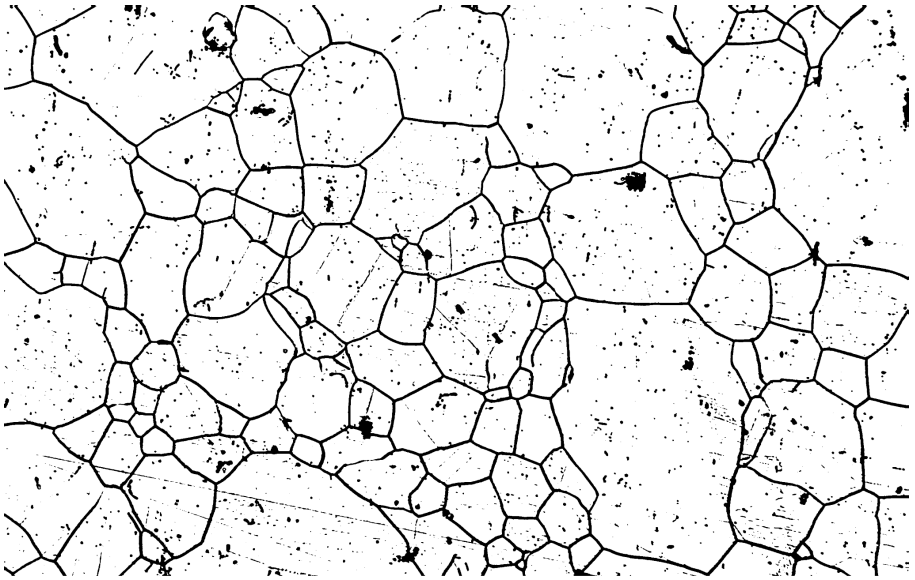
Topic:Development of Python Library for Stereology

Mentor : Prof. Sandeep Sangal
Name:Kanhaiya Kumar
Roll no.-200487

Work done:

1)Real microstructure:



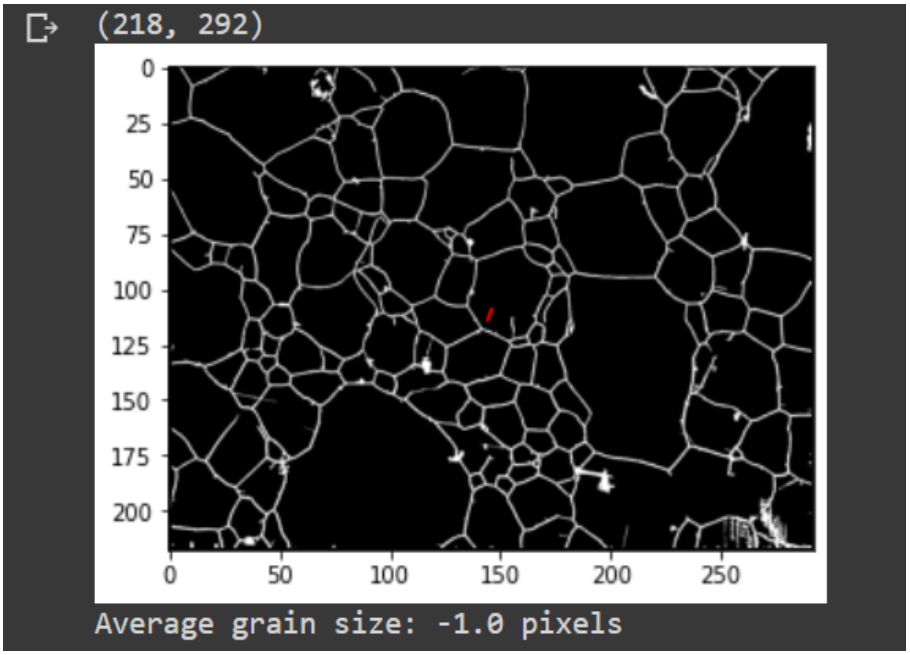
2)

```
import cv2
# Load the microstructure image
grain_image = cv2.imread('/content/sample.png',0)

# Calculate the average grain size using the intercept method
avg_grain_size = intercept_method(grain_image,20,5)

print(f"Average grain size: {avg_grain_size} pixels")
```

Output:



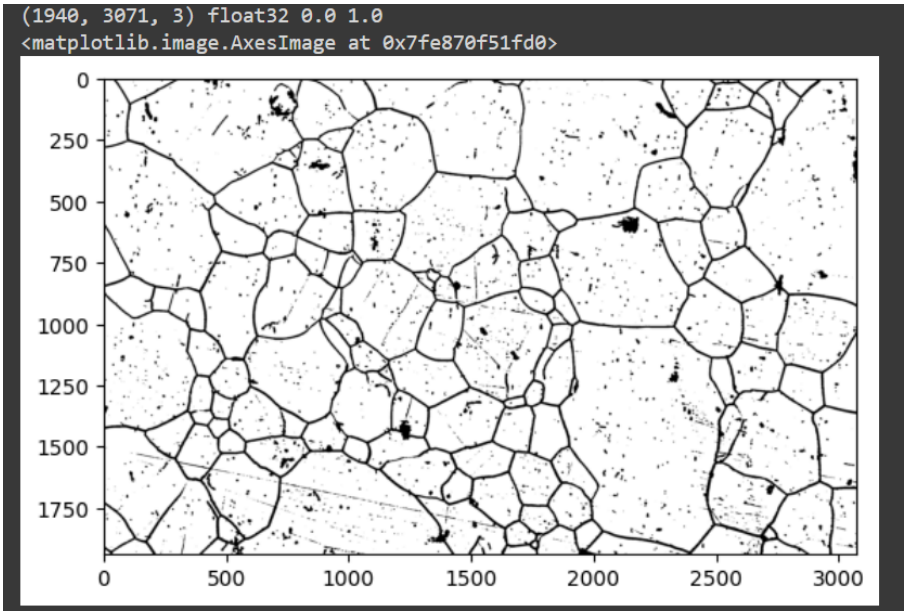
3)Importing Libraries

```
import matplotlib.pyplot as plt
import scipy.ndimage as nd
import matplotlib.image as img
import numpy as np
from skimage import color, filters, morphology, measure, draw
```

4)Image display with its properties

```
image = plt.imread('/content/download (7).png')
print(image.shape, image.dtype, image.min(), image.max())
plt.imshow(image,cmap='gray')
```

Output

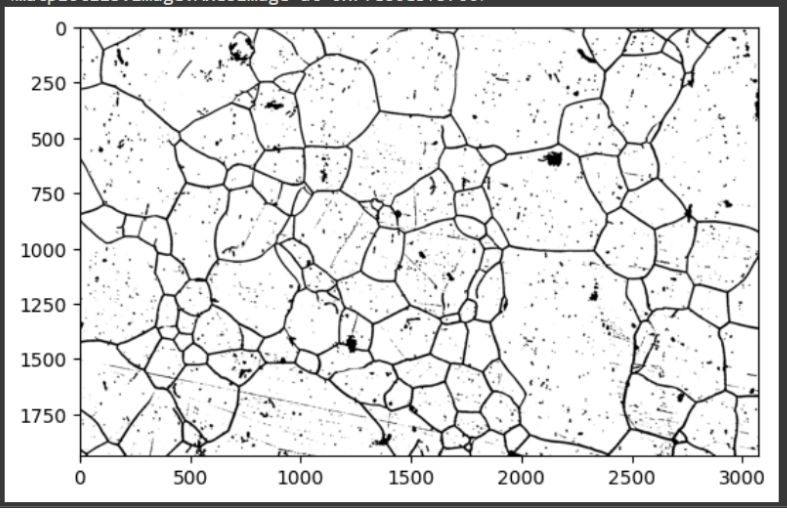


5)Changing RGB to Grayscale

```
[ ]  image = image[:,:,0:3]
     gray = 255*color.rgb2gray(image)
     gray = gray.astype(dtype='uint8')
     print(gray.shape, gray.dtype, gray.min(), gray.max())
     plt.imshow(gray,cmap='gray')
```

Output

```
(1940, 3071) uint8 0 254
<matplotlib.image.AxesImage at 0x7fe86ebf5760>
```



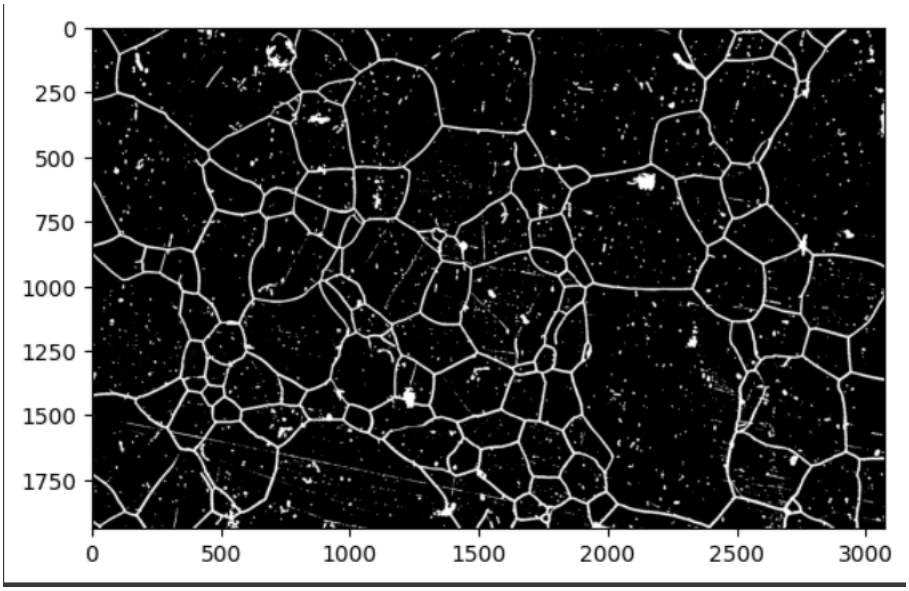
6)Binarising

```
[ ]  freq_hist = nd.histogram(gray,0,255,256)
     bins = [i for i in range(256)]
     plt.bar(bins, freq_hist)
```

```
[ ]  t = 170
     t = filters.threshold_otsu(gray)
     print(t)
     binary = gray <= t
     print(binary)
     plt.imshow(binary, cmap='gray')
```

Output

```
0
[[False False False ... False False False]
 [False False False ... False False False]
 [False False False ... False False False]
 ...
 [False False False ... False False False]
 [False False False ... False False False]
 [False False False ... False False False]]
<matplotlib.image.AxesImage at 0x7fe86d8d77c0>
```

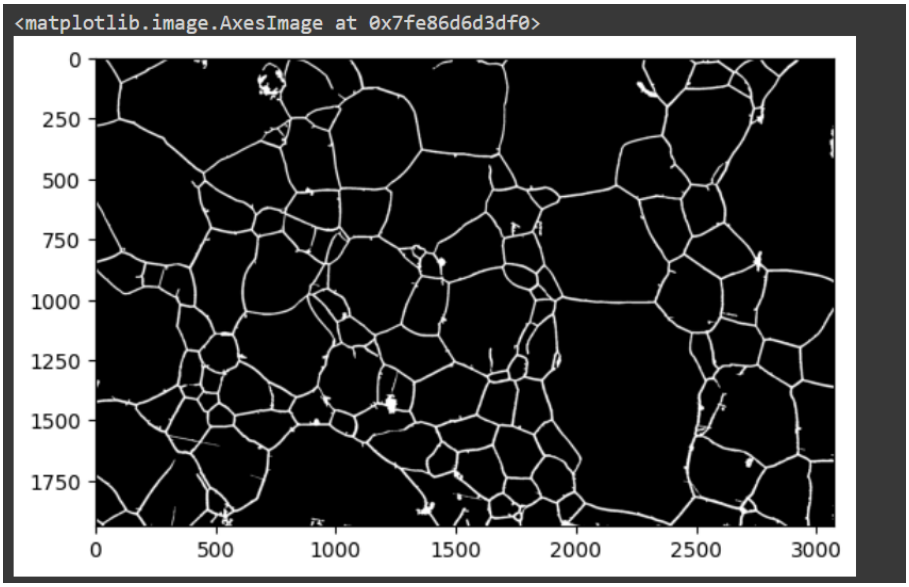


7)Dilation

```
seed = binary.copy()
seed[1:-1,1:-1] = False
mask = binary.copy()
seed = morphology.reconstruction(seed, mask, method = 'dilation')
plt.imshow(seed, cmap = 'gray')
```

Output

```
<matplotlib.image.AxesImage at 0x7fe86d6d3df0>
```

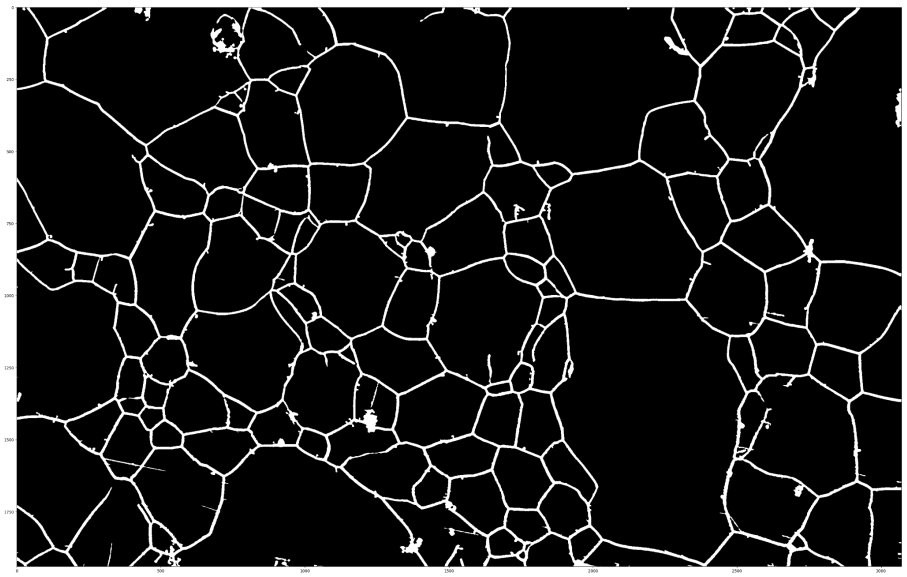


8)Skeletonization

```
[ ]  thinned = morphology.skeletonize(seed)
     print(type(thinned))
     print(thinned)
     plt.figure(figsize = (40,40))
     plt.imshow(seed, cmap='gray')
```

Output

```
<class 'numpy.ndarray'>
[[False False False ... False False False]
 [False False False ... False False False]
 [False False False ... False False False]
 ...
 [False False False ... False False False]
 [False False False ... False False False]
 [False False False ... False False False]]
<matplotlib.image.AxesImage at 0x7fe86d746df0>
```



9)Thinning and gridlines

```
[ ]  grid = np.zeros(thinned.shape, dtype = 'bool')
     delta = int(np.rint(grid.shape[0]/4))
     print(delta)
     n = 0
     for i in range(delta, grid.shape[0],delta):
         n += 1
         grid[i,0:-1] = True
     print(n)
     plt.figure(figsize=(40,40))
     plt.imshow(thinned + grid, cmap = 'gray')
```

Output

10)Display of intersection

```
[ ]  intersection1 = np.logical_and(seed,grid)
     intersection2 = np.logical_and(thinned,grid)
     plt.figure(figsize=(40,40))
     plt.imshow(intersection1, cmap='gray')
```

Output

11)Counting the Number of intersections

```
[ ]  labels1 = morphology.label(intersection1)
     labels2 = morphology.label(intersection2)
     props1 = measure.regionprops(labels1)
     props2 = measure.regionprops(labels2)
     print (str(len(props1)) + '  intersection coordinates from image prior to thinning')
     #for i in range(len(props1)):
     #    print(props1[i].centroid)
     print (str(len(props2)) + '  intersection coordinates from image after thinning')
     #for i in range(len(props2)):
     #    print(props2[i].centroid)
```

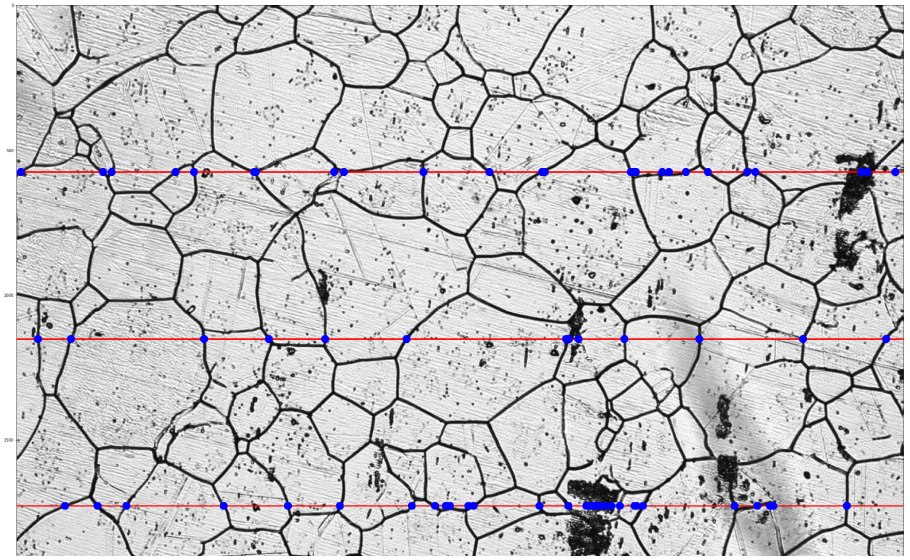
Output

```
48  intersection coordinates from image prior to thinning
47  intersection coordinates from image after thinning
```

12)Pointing out the intersection points on original microstructure

```
image2 = image.copy()
delta = int(np.rint(grid.shape[0]/4))
n = 0
for i in range(delta, image2.shape[0],delta):
    n += 1
    image2[i-2:i+3,0:-1,0] = 255
    image2[i-2:i+3,0:-1,1] = 0
    image2[i-2:i+3,0:-1,2] = 0
r = 14
for i in range(len(props2)):
    rr, cc = draw.circle.perimeter(int(props2[i].centroid[0]), int(props2[i].centroid[1]),r)
    image2[rr, cc, :] = [0, 0, 255]
plt.figure(figsize=(40,40))
plt.imshow(image2)
```

output



Link to code:
https://colab.research.google.com/drive/16FUKkEwVee0C9H39iLmSPp5rgEfreWCS#scrollTo=0MS0Lpt76wgE