In [1]:
```
import cv2
from scipy import ndimage
from skimage import io, color, measure, exposure, img_as_ubyte
import numpy as np
import h5py
import hdf5plugin
import os
import matplotlib.pyplot as plt
import napari
import pyclesperanto_prototype as cle   #https://github.com/clEsperanto/pyclespe
import skimage.measure as skm
from scipy.spatial import Voronoi
from scipy.ndimage import zoom
```

In [2]:
```
h5name = 'C:/Users/AO0736/Box/Indrajeet/LabDCT/AluFoams_3p2mm_bigscan_30s_export

with h5py.File(h5name, "r") as hin:
    tomo = hin['AbsorptionCT']['Data'][:]
    labdct = hin['LabDCT']['Data']['GrainId'][:]*hin['LabDCT']['Data']['Mask'][:
```

In [3]:
```
labdct.shape
```

Out[3]:
```
(310, 332, 333)
```

In [4]:
```
zoom_factors = [c/g for c,g in zip(tomo.shape,labdct.shape)]
print(zoom_factors)
labdct_zoom=zoom(labdct,zoom_factors,order=0)
print(tomo.shape)
print(labdct_zoom.shape)
```

```
[2.9612903225806453, 2.960843373493976, 2.960960960960961]
(918, 983, 986)
(918, 983, 986)
```

In [5]:
```
labdct_zoom = labdct_zoom[3*100:3*200, 3*100:3*200, :]

tomo = tomo[3*100:3*200, 3*100:3*200, :]
```

In [6]:
```
print(labdct_zoom.shape)
print(tomo.shape)
```

```
(300, 300, 986)
(300, 300, 986)
```

In [7]:
```
import pandas as pd
props = skm.regionprops(labdct_zoom)
pd.DataFrame(props)
```

Out[7]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|---|
| **0** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **1** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **2** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **3** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **4** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **5** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **6** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **7** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **8** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **9** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **10** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **11** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **12** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **13** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **14** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **15** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **16** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **17** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **18** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |
| **19** | area | area_bbox | area_convex | area_filled | axis_major_length | axis_minor_length | bbox | c |

20 rows × 30 columns

In [8]:

```
points = np.zeros((3,20))

for i in range (20):
    x_pos = props[i]['centroid'][0]
    x_pos = np.asarray(x_pos)

    y_pos = props[i]['centroid'][1]
    y_pos = np.asarray(y_pos)

    z_pos = props[i]['centroid'][2]
    z_pos = np.asarray(z_pos)

    points[0,i]= x_pos
    points[1,i]= y_pos
    points[2,i]= z_pos

    #print(a,b,c)
```

```
print(points.shape)
```

```
(3, 20)
```

In [9]:
```
# selet a GPU with the following in the name. This will fallback to any other GP
cle.select_device('RTX')
```

Out[9]:
```
<NVIDIA RTX A2000 Laptop GPU on Platform: NVIDIA CUDA (1 refs)>
```

In [ ]:
```

```

In [10]:
```
image = cle.create(tomo.shape)
cle.set(image, 0)

spots = cle.pointlist_to_labelled_spots(points, image)

#spots = cle.maximum_sphere(spots, radius_x=2, radius_y=2, radius_z=2)
```

In [ ]:
```

```

In [11]:
```
spots.shape
```

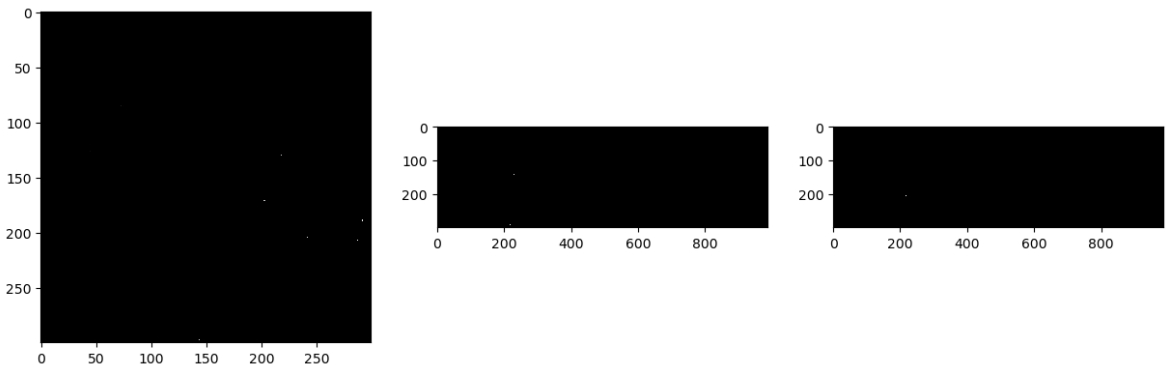
Out[11]:
```
(300, 300, 986)
```

In [12]:
```
def show(image_to_show, labels=False):
    """
    This function generates three projections: in X-, Y- and Z-direction and sho
    """
    projection_x = cle.maximum_x_projection(image_to_show)
    projection_y = cle.maximum_y_projection(image_to_show)
    projection_z = cle.maximum_z_projection(image_to_show)

    fig, axs = plt.subplots(1, 3, figsize=(15, 15))
    cle.imshow(projection_x, plot=axs[0], labels=labels)
    cle.imshow(projection_y, plot=axs[1], labels=labels)
    cle.imshow(projection_z, plot=axs[2], labels=labels)

show(spots)
print(spots.shape)
```

```
(300, 300, 986)
```




In [ ]:
```
#spots_image = cle.pull(spots)
```

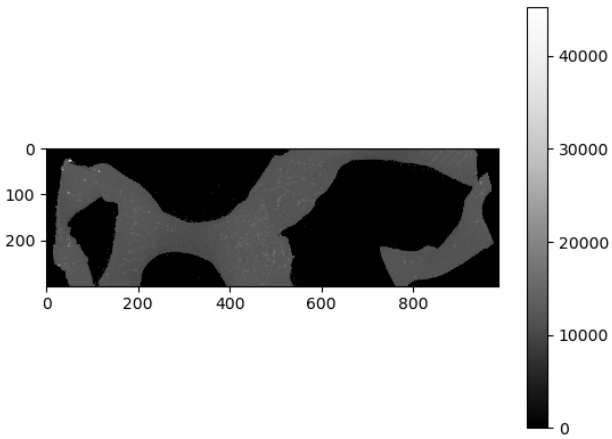
In [ ]:
```
#viewer = napari.view_image(tomo_compress)
#viewer = napari.view_image(spots_image)
```

In [13]:
```
tom = tomo
```

In [14]:
```
background = 7000
tom[tom<background] = 0
```

In [15]:
```
tom_gpu = cle.push(tom)
tom_gpu
```

Out[15]:




**cle._ image**

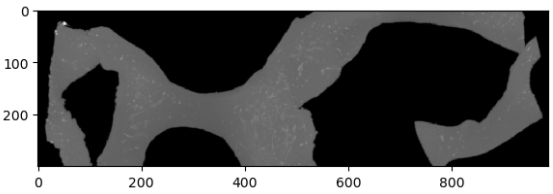
| | |
|---|---|
| shape | (300, 300, 986) |
| dtype | float32 |
| size | 338.5 MB |
| min | 0.0 |
| max | 45307.0 |

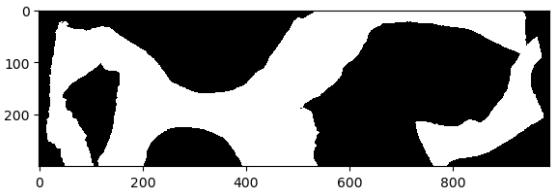
In [16]:
```
sigma_outline = 1


blurred = cle.gaussian_blur(tom, sigma_x=sigma_outline, sigma_y=sigma_outline, s

binary = cle.threshold_otsu(blurred)

fig, axs = plt.subplots(1, 2, figsize=(15, 15))
cle.imshow(blurred, plot=axs[0])
cle.imshow(binary, plot=axs[1])
```

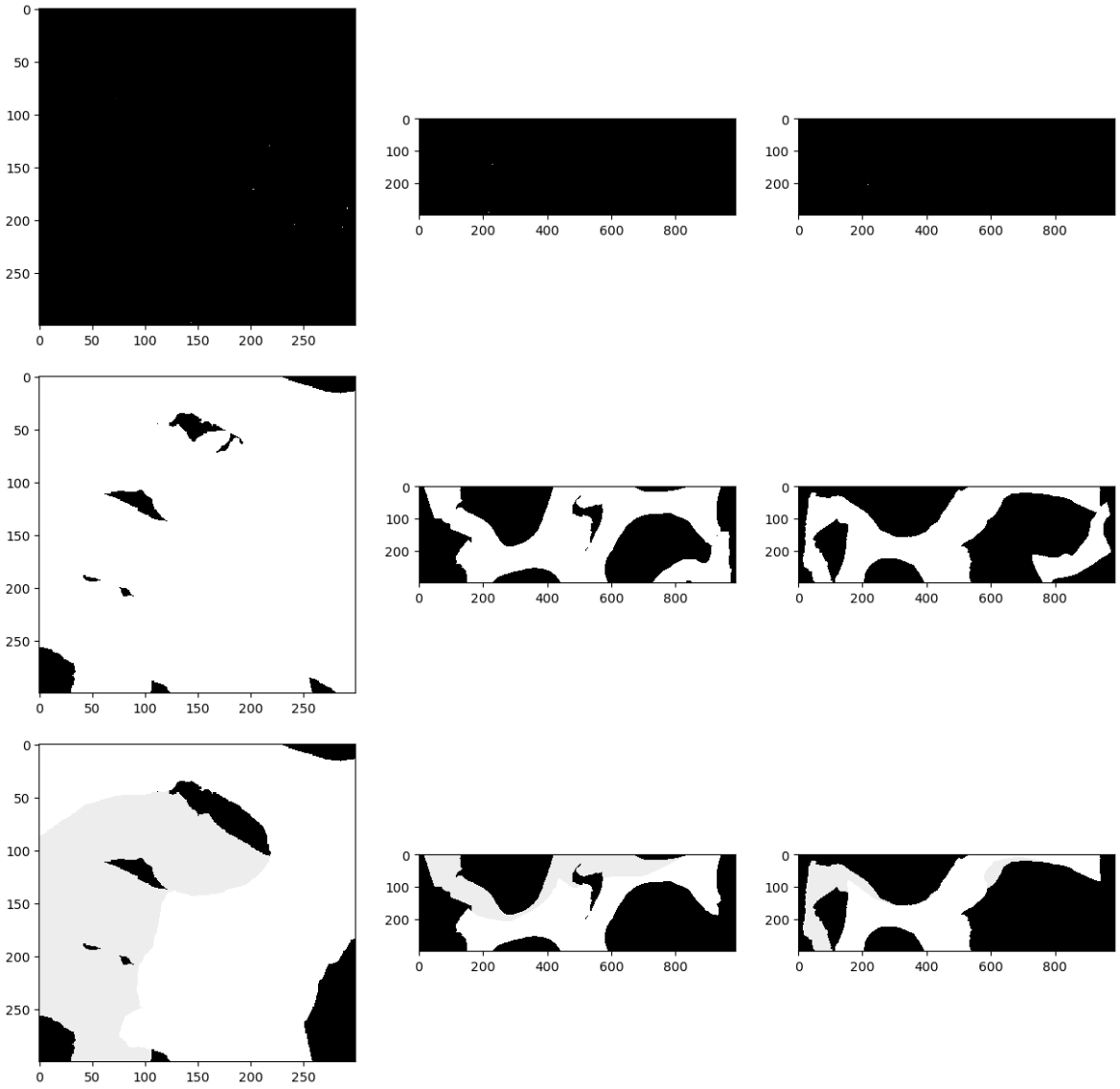



In [ ]:
```
```

In [17]:
```
voronoi_diagram = cle.masked_voronoi_labeling(spots, binary)

show(spots)
show(binary)
show(voronoi_diagram)
```



In [ ]:

In [18]:
```
output_gpu = cle.pull(voronoi_diagram)
#output_gpu
```

In [19]:
```
viewer = napari.view_image(tomo)
viewer.add_labels(output_gpu)
viewer.add_points(np.transpose(points), size=5)
```

Out[19]: <Points layer 'Points' at 0x19fffa9ee20>

In [ ]: