In [3]:

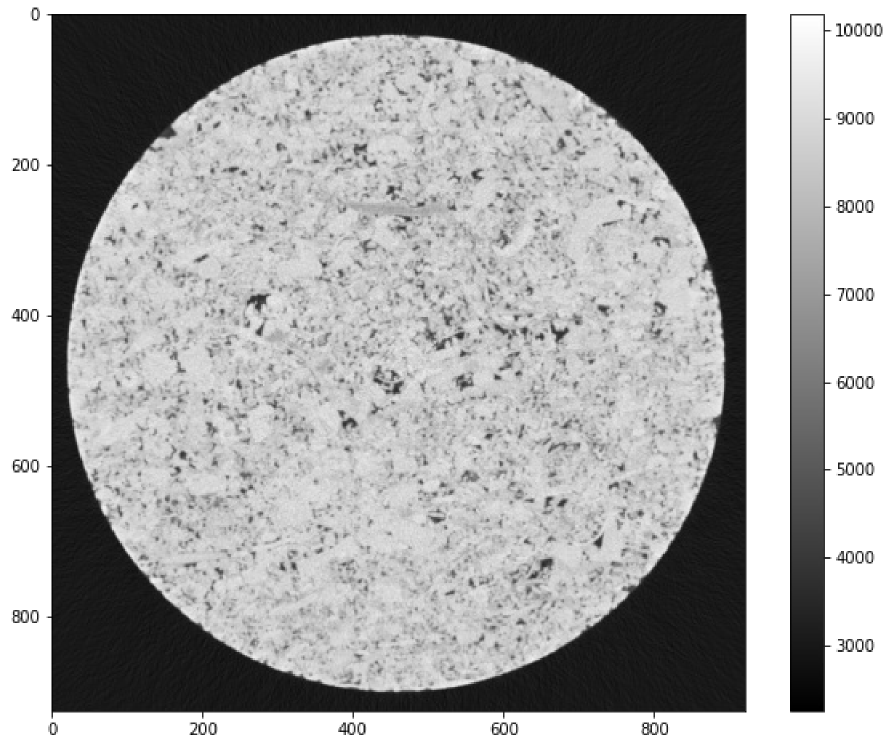
```
import numpy as np
import matplotlib.pyplot as plt

from skimage import io
```

In [7]:

```
# Try to open an example image
example_image = "tiff/antes0900.tiff"
e_img = io.imread(example_image)

plt.figure(figsize=(10, 8))
plt.imshow(e_img, cmap="gray")
plt.colorbar()
plt.show()
```



In [8]:

```
# print the shape of the image so we can construct our np array with all the images volume
print(e_img.shape)
```

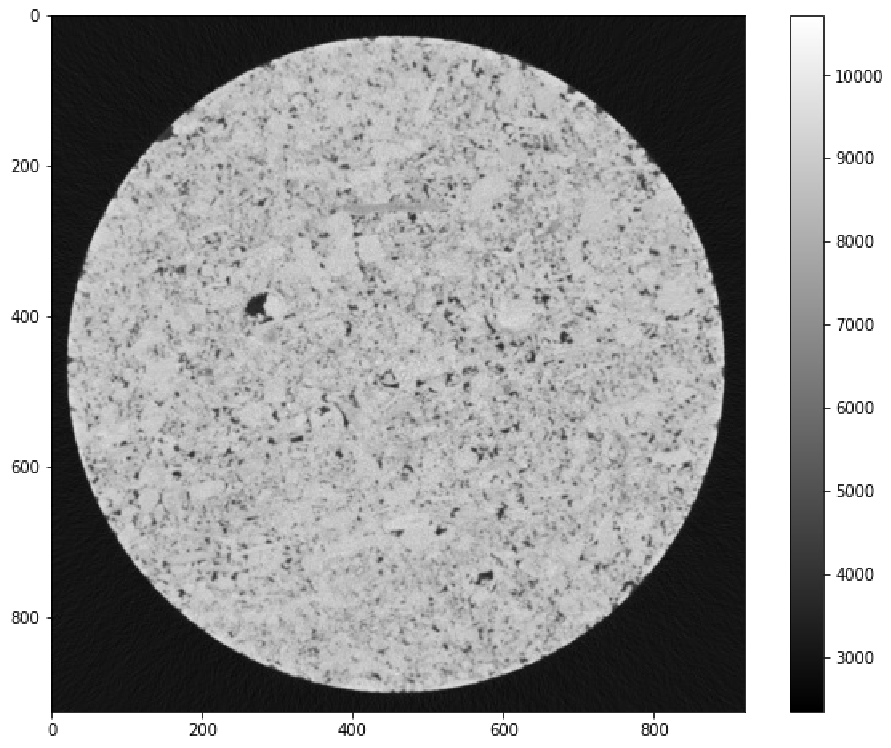
```
(925, 920)
```

In [13]:

```
# creating our volume of tiff images
root = "tiff/antes09"
vol = np.zeros((13, 925, 920), dtype="uint16")
for i in range(0, 13):
    if i < 10:
        fname = root + "0" + str(i) + ".tiff"
    else:
        fname = root + str(i) + ".tiff"
    vol[i,:, :] = io.imread(fname)
```

In [19]:

```
# show some tiff image from the volume
plt.figure(figsize=(10,8))
plt.imshow(vol[12,:], cmap="gray")
plt.colorbar()
plt.show()
```



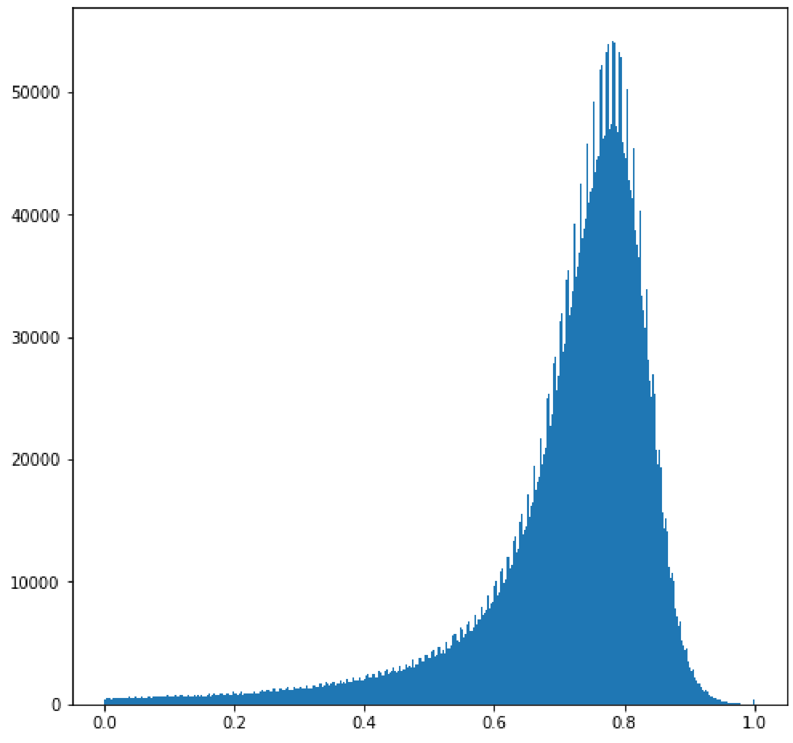
In [20]:

```
# before printing the data histogram we should normalize the data
def scale(data, x_min, x_max, nx_min, nx_max):
    data = (data - x_min) * ((nx_max-nx_min)/(x_max-x_min))+nx_min
    # generalizing values that are too high or too low
    data[data>nx_max] = nx_max
    data[data<nx_min] = nx_min
    return data

vol_f = scale(vol.astype(np.float32), 4000, 10000, 0, 1)
```

In [23]:

```
# ignore 0 values so we can better observe values >0 in the histogram
data = vol_f[vol_f>0]
plt.figure(figsize=(8,8))
n, bins, patches = plt.hist(data.ravel(), bins="auto")
plt.show()
```



In [ ]: