In [387…
```
import numpy as np
import skimage
import skimage.io
import matplotlib.pyplot as plt
```

In [388…
```
image = skimage.io.imread('https://cildata.crbs.ucsd.edu/media/images/13901/13901.tif')
```

In [389…
```
image
```

Out[389]:
```
array([[219, 225, 220, ..., 256, 258, 272],
       [218, 211, 217, ..., 270, 268, 278],
       [217, 217, 214, ..., 263, 268, 262],
       ...,
       [337, 343, 321, ..., 266, 287, 285],
       [320, 323, 339, ..., 268, 295, 281],
       [329, 353, 335, ..., 276, 281, 291]], dtype=uint16)
```

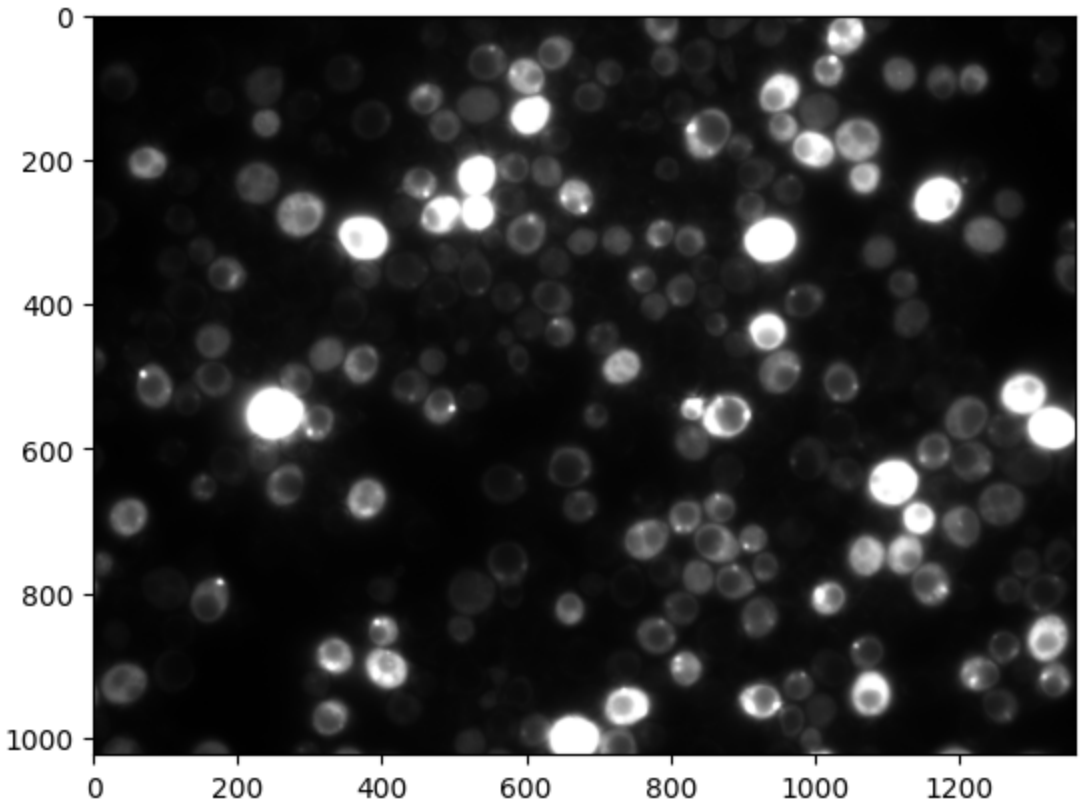
In [390…
```
type(image)
```

Out[390]:
```
numpy.ndarray
```

In [391…
```
image.shape
```

Out[391]:
```
(1024, 1360)
```

In [392…
```
plt.imshow(image, cmap='gray');
```



In [393…
```
image
```

Out[393]:
```
array([[219, 225, 220, ..., 256, 258, 272],
       [218, 211, 217, ..., 270, 268, 278],
       [217, 217, 214, ..., 263, 268, 262],
       ...,
       [337, 343, 321, ..., 266, 287, 285],
```

```
        [320, 323, 339, ..., 268, 295, 281],
        [329, 353, 335, ..., 276, 281, 291]], dtype=uint16)
```

In [394…
```
image[0,0]
```

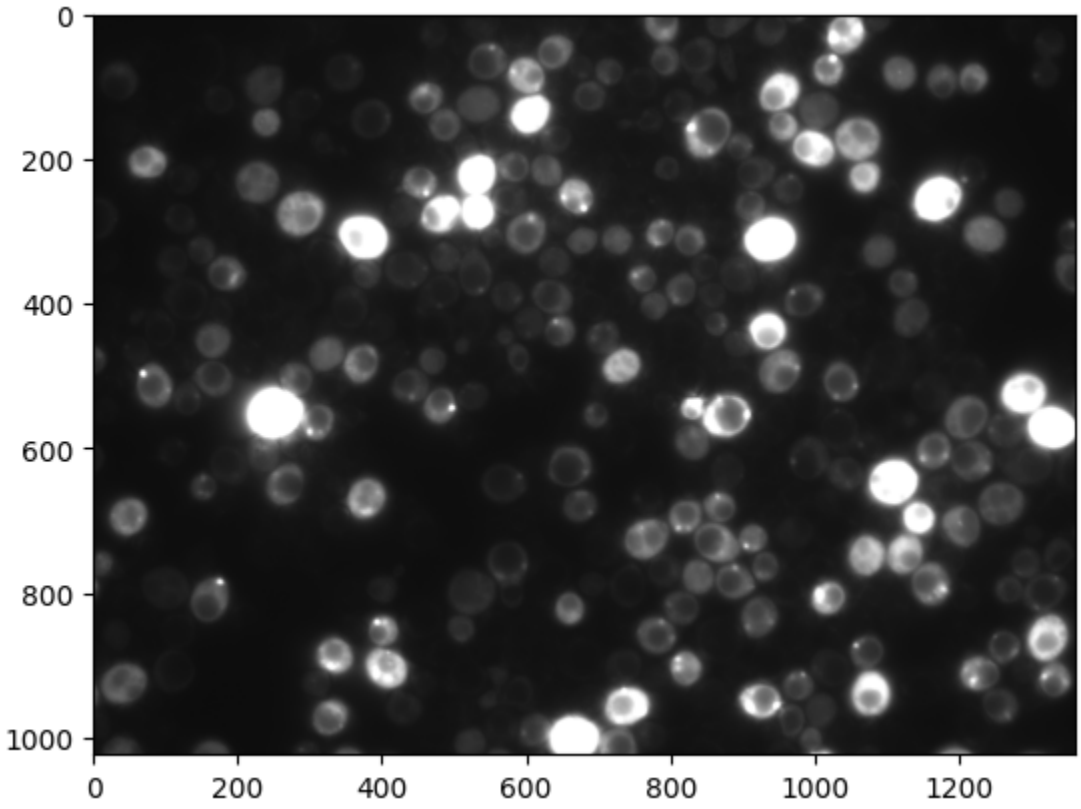
Out[394]:
```
219
```

In [395…
```
image[0,0] = 2
```

In [396…
```
image
```

Out[396]:
```
array([[  2, 225, 220, ..., 256, 258, 272],
       [218, 211, 217, ..., 270, 268, 278],
       [217, 217, 214, ..., 263, 268, 262],
       ...,
       [337, 343, 321, ..., 266, 287, 285],
       [320, 323, 339, ..., 268, 295, 281],
       [329, 353, 335, ..., 276, 281, 291]], dtype=uint16)
```

In [397…
```
plt.imshow(image, cmap='gray');
```

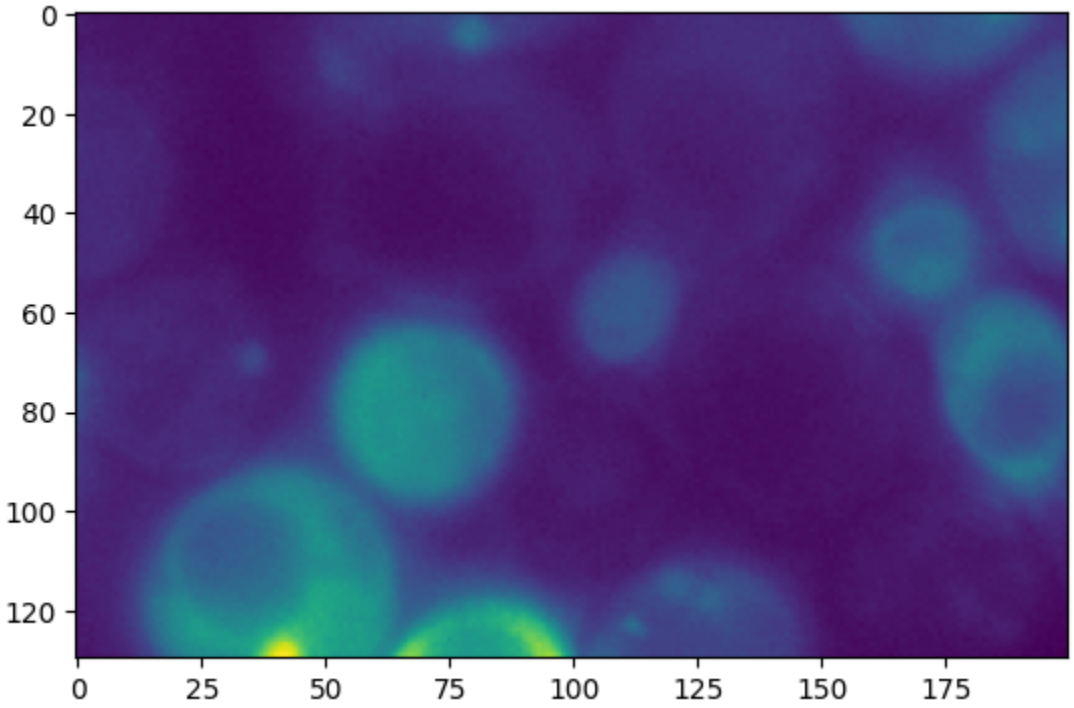


In [398…
```
image[400:530, 400:600]
```

Out[398]:
```
array([[375, 352, 370, ..., 470, 452, 469],
       [372, 368, 365, ..., 462, 458, 472],
       [381, 370, 359, ..., 478, 461, 480],
       ...,
       [349, 365, 346, ..., 287, 285, 290],
       [333, 349, 340, ..., 288, 276, 275],
       [355, 351, 337, ..., 282, 274, 267]], dtype=uint16)
```

In [399…
```
cropped = image[400:530, 400:600].copy()
```

In [400…
```
plt.imshow(cropped);
```



In [401…
```
cropped.shape
```
Out[401]:
```
(130, 200)
```

In [402…
```
image[400:530, :].shape
```
Out[402]:
```
(130, 1360)
```

In [403…
```
image[400:, :].shape
```
Out[403]:
```
(624, 1360)
```

In [404…
```
np.zeros(shape=(5,3))
```
Out[404]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [405…
```
np.ones(shape=(5,3))
```
Out[405]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

In [406…
```
np.arange(start=4, stop=12, step=2)
```
Out[406]:
```
array([ 4,  6,  8, 10])
```

In [407…
```
np.random.poisson(lam=3, size=(3,5))
```
Out[407]:
```
array([[1, 1, 3, 0, 2],
       [2, 6, 0, 3, 2],
       [3, 3, 1, 5, 8]])
```

In [408…
```
cropped_plus_three = cropped + 3
```

In [409…
```
cropped_plus_three
```

Out[409]:
```
array([[378, 355, 373, ..., 473, 455, 472],
       [375, 371, 368, ..., 465, 461, 475],
       [384, 373, 362, ..., 481, 464, 483],
       ...,
       [352, 368, 349, ..., 290, 288, 293],
       [336, 352, 343, ..., 291, 279, 278],
       [358, 354, 340, ..., 285, 277, 270]], dtype=uint16)
```

In [410…
```
cropped
```

Out[410]:
```
array([[375, 352, 370, ..., 470, 452, 469],
       [372, 368, 365, ..., 462, 458, 472],
       [381, 370, 359, ..., 478, 461, 480],
       ...,
       [349, 365, 346, ..., 287, 285, 290],
       [333, 349, 340, ..., 288, 276, 275],
       [355, 351, 337, ..., 282, 274, 267]], dtype=uint16)
```

In [411…
```
cropped * 3
cropped / 3
cropped - 3
```

Out[411]:
```
array([[372, 349, 367, ..., 467, 449, 466],
       [369, 365, 362, ..., 459, 455, 469],
       [378, 367, 356, ..., 475, 458, 477],
       ...,
       [346, 362, 343, ..., 284, 282, 287],
       [330, 346, 337, ..., 285, 273, 272],
       [352, 348, 334, ..., 279, 271, 264]], dtype=uint16)
```

In [412…
```
a = 3
```

In [413…
```
a ** 2
```

Out[413]:
```
9
```

In [414…
```
cropped ** 2
```

Out[414]:
```
array([[ 9553, 58368,  5828, ..., 24292,  7696, 23353],
       [ 7312,  4352,  2153, ..., 16836, 13156, 26176],
       [14089,  5828, 63345, ..., 31876, 15913, 33792],
       ...,
       [56265,  2153, 54180, ..., 16833, 15689, 18564],
       [45353, 56265, 50064, ..., 17408, 10640, 10089],
       [60489, 57665, 48033, ..., 13988,  9540,  5753]], dtype=uint16)
```

In [415…
```
np.cos(cropped)
```

Out[415]:
```
array([[-0.40805453,  0.98998827,  0.7597075 , ...,  0.32583833,
         0.925159  , -0.61949694],
       [ 0.27513435, -0.9074341 ,  0.8390551 , ..., -0.982774  ,
         0.7822497 ,  0.7240764 ],
       [-0.64689636,  0.7597075 ,  0.6536208 , ...,  0.88795507,
        -0.6865085 , -0.7877331 ],
       ...,
       [-0.96016186,  0.8390551 ,  0.91111785, ..., -0.44011596,
        -0.63334256,  0.5624289 ],
       [ 0.9999611 , -0.96016186,  0.75966835, ...,  0.51779556,
         0.89598435,  0.11041721],
       [-1.        ,  0.65366644, -0.66029406, ...,  0.73621315,
        -0.776667  , -0.99937433]], dtype=float32)
```

In [416…
```
np.max(cropped)
```

Out[416]: 1568

In [417…
```
np.min(cropped)
```

Out[417]: 267

In [418…
```
np.std(cropped)
```

Out[418]: 172.53921546746614

In [419…
```
172.53921546746614
```

Out[419]: 172.53921546746614

In [420…
```
np.max(cropped)
```

Out[420]: 1568

In [421…
```
cropped.max()
```

Out[421]: 1568

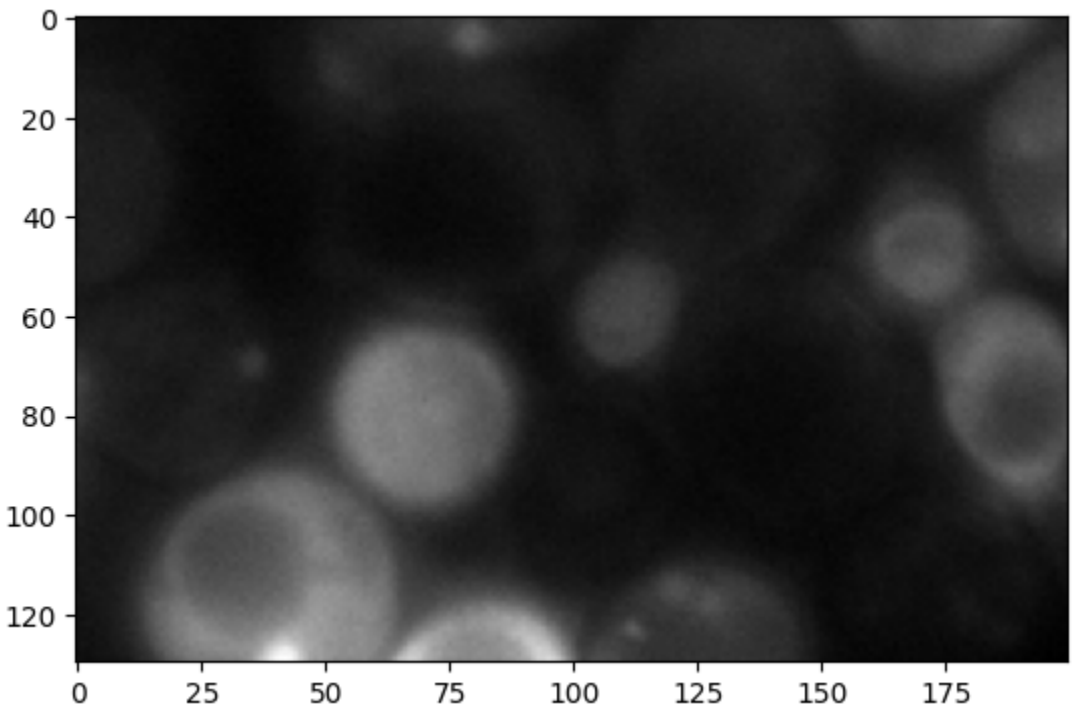
In [422…
```
cropped.shape
```

Out[422]: (130, 200)

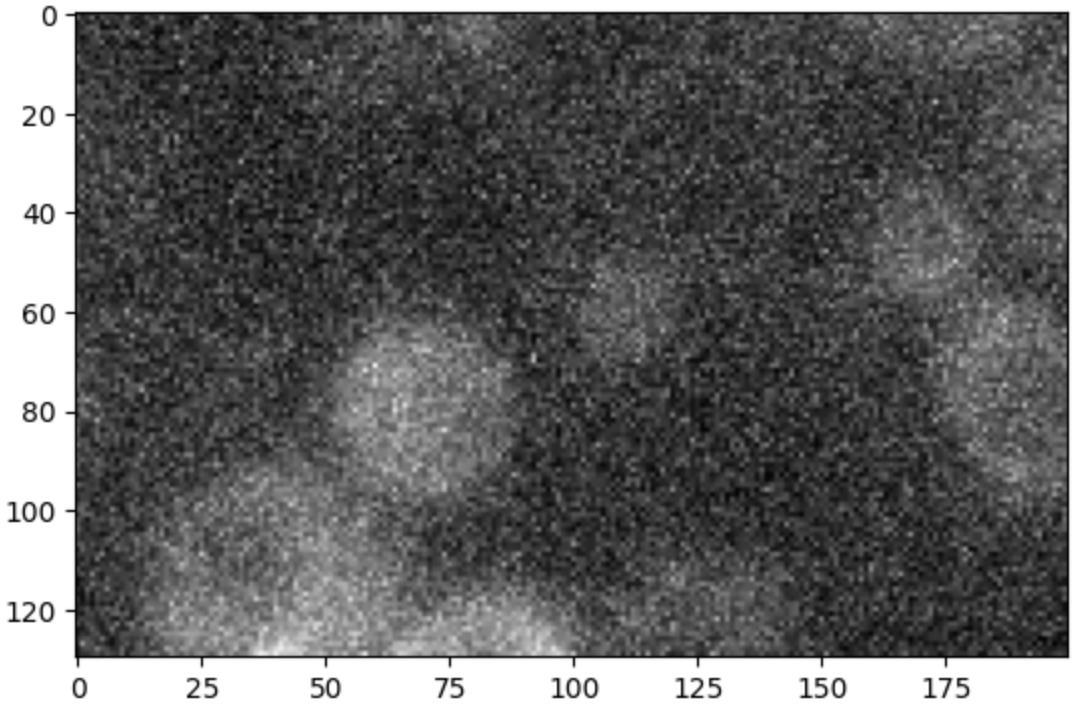
In [423…
```
noise_image = np.random.poisson(lam=3, size=cropped.shape)
```

In [424…
```
corrupted = cropped + 100*noise_image
```

In [425…
```
plt.imshow(cropped, cmap='gray');
```



In [426…
```
plt.imshow(corrupted, cmap='gray');
```



In [427…
```
a = 2
```

In [428…
```
a > 3
```

Out[428]:
```
False
```

In [429…
```
a < 3
```

Out[429]:
```
True
```

In [430…
```
out = a > 3
out
```

Out[430]:
```
False
```

In [431…
```
type(out)
```

Out[431]:
```
bool
```

In [432…
```
cropped
```

Out[432]:
```
array([[375, 352, 370, ..., 470, 452, 469],
       [372, 368, 365, ..., 462, 458, 472],
       [381, 370, 359, ..., 478, 461, 480],
       ...,
       [349, 365, 346, ..., 287, 285, 290],
       [333, 349, 340, ..., 288, 276, 275],
       [355, 351, 337, ..., 282, 274, 267]], dtype=uint16)
```

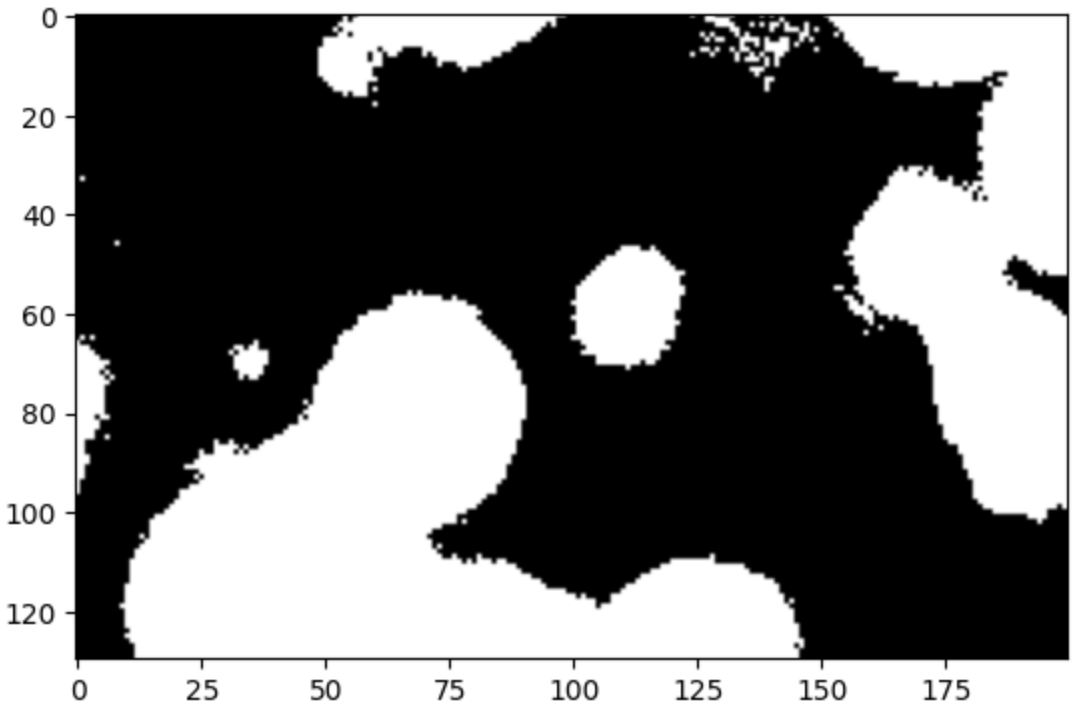
In [433…
```
cropped > 450
```

Out[433]:
```
array([[False, False, False, ...,  True,  True,  True],
       [False, False, False, ...,  True,  True,  True],
       [False, False, False, ...,  True,  True,  True],
       ...,
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False]])
```

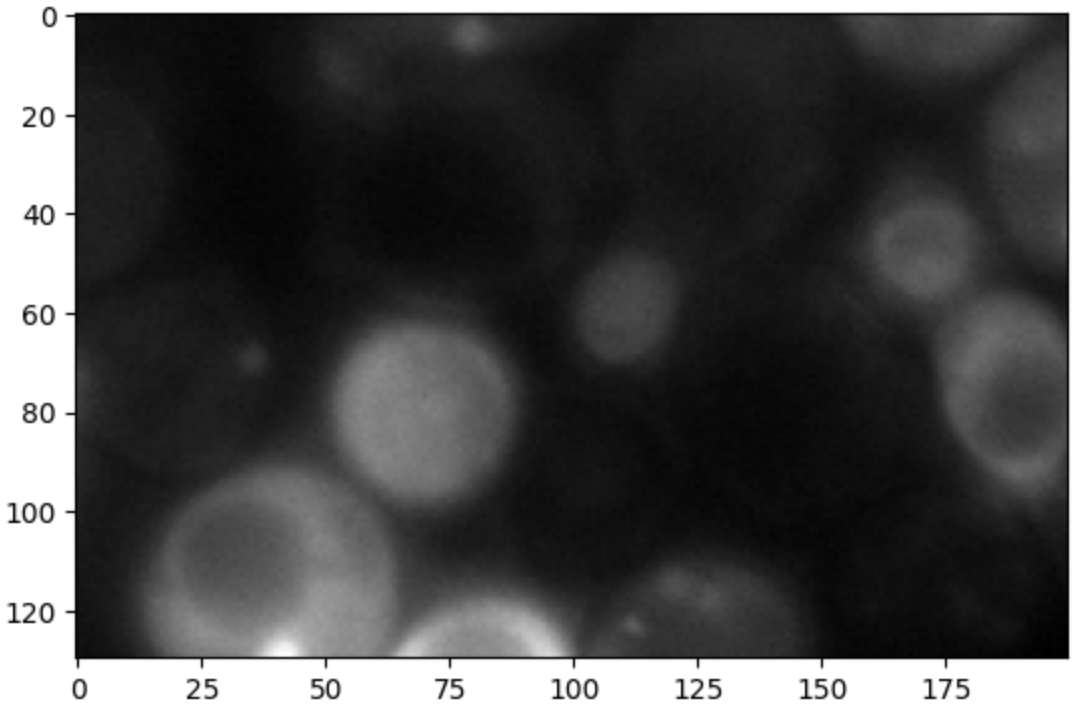
In [434…
```
array_bool = cropped > 450
```

In [435…
```
plt.imshow(array_bool, cmap='gray')
```

Out[435]:
```
<matplotlib.image.AxesImage at 0x1d0912de750>
```

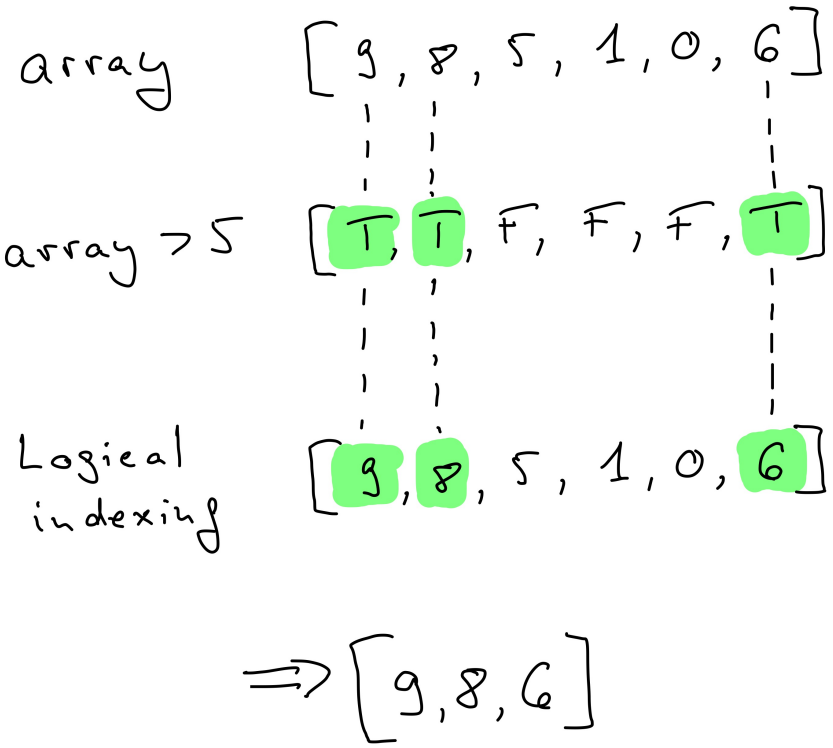


In [436…
```
plt.imshow(cropped, cmap='gray');
```



In [437…
```
from IPython.display import Image
Image(url='https://github.com/guiwitz/ISDAwPython_day2/raw/master/images/logical_indexin
```

Out[437]:



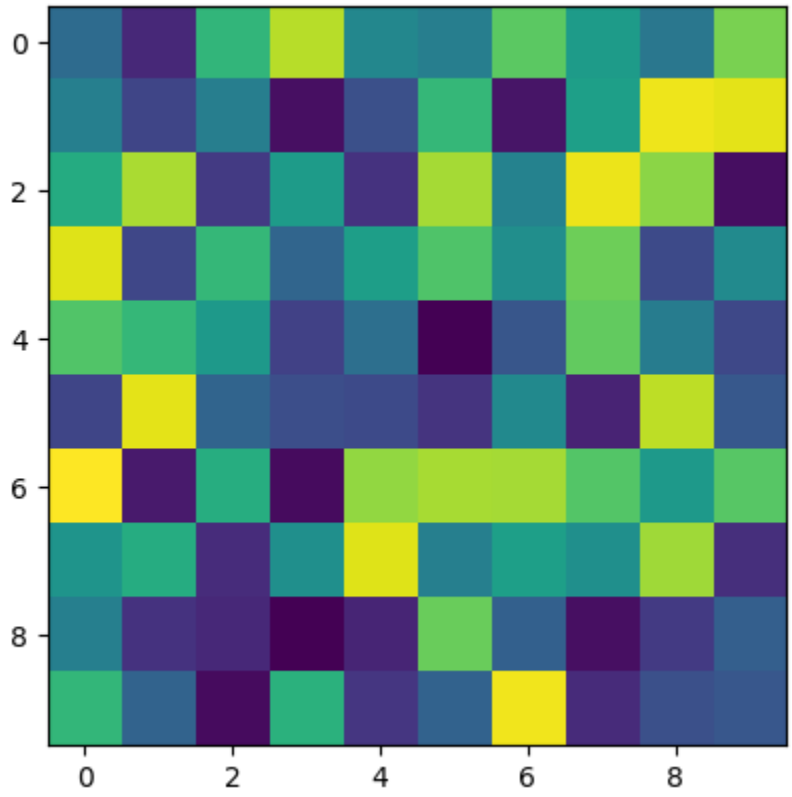
In [438…
```
cropped[array_bool]
```

Out[438]:
```
array([468, 466, 465, ..., 496, 488, 472], dtype=uint16)
```

In [439…
```
extracted_pixels = cropped[array_bool]
extracted_pixels.mean()
```

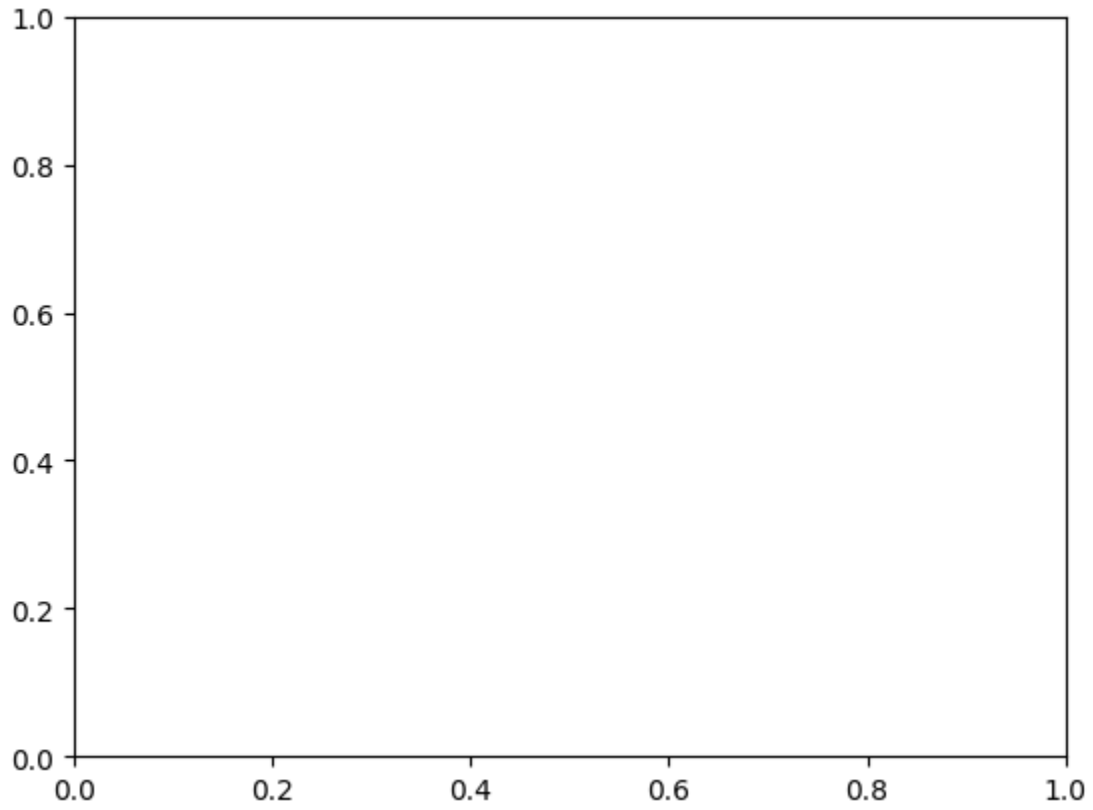
Out[439]:
```
658.8238455632335
```

In [440…
```
import numpy as np
plt.imshow(np.random.randint(0,255,(10,10)));
```



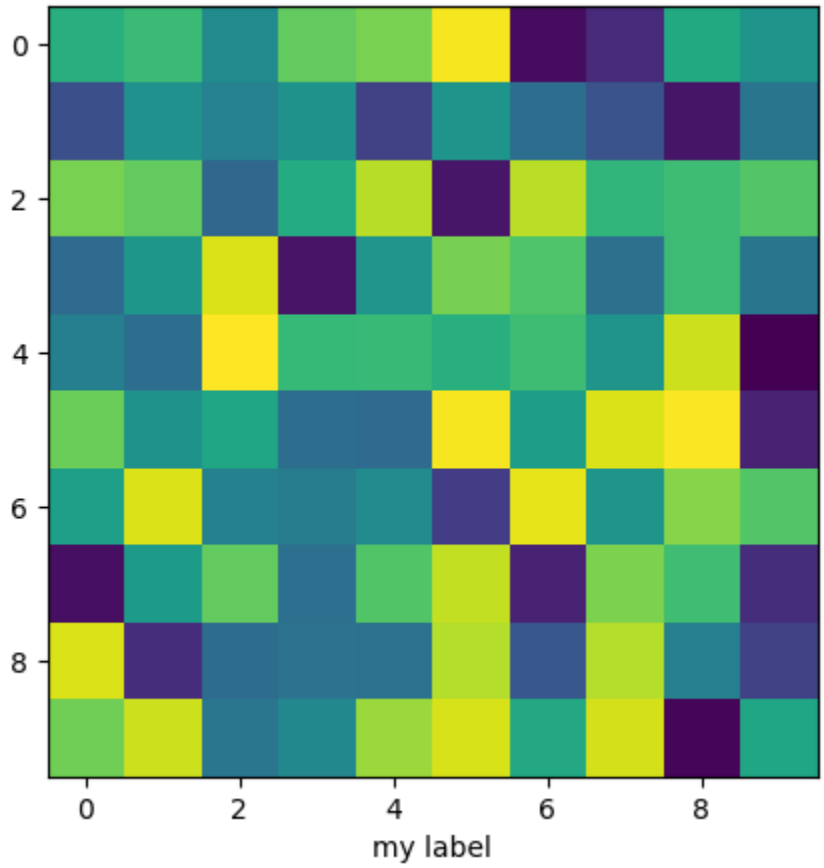
In [441…

```
fig, ax = plt.subplots()
```



In [442…

```
fig, ax = plt.subplots(figsize=(5,5))
ax.imshow(np.random.randint(0,255,(10,10)));
ax.set_xlabel('my label');
```



In [443…
```
import skimage.io
image = skimage.io.imread('https://cildata.crbs.ucsd.edu/media/images/13901/13901.tif')
```

In [444…
```
image.shape
```

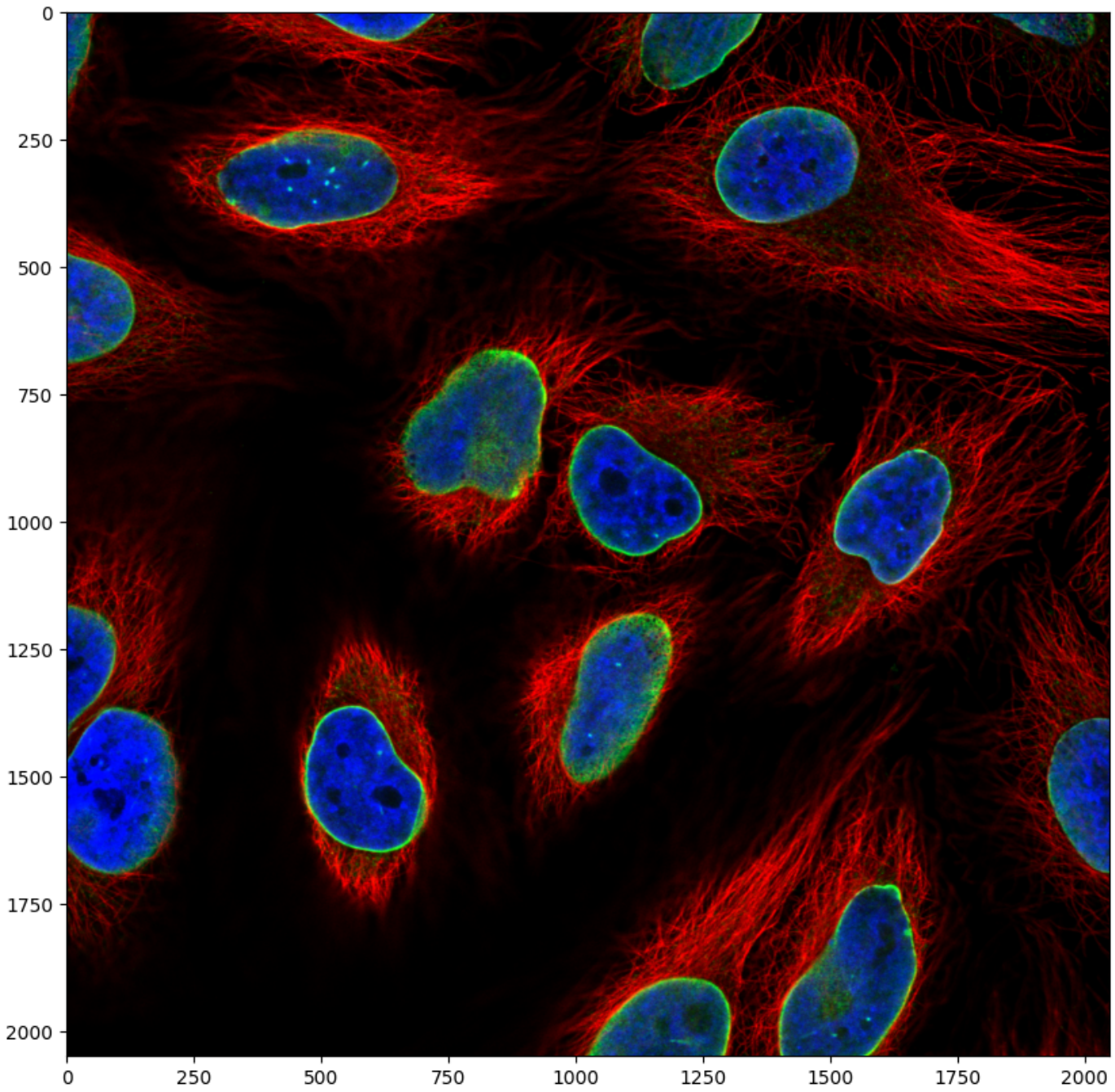
Out[444]:
```
(1024, 1360)
```

In [445…
```
image_rgb = skimage.io.imread('https://github.com/guiwitz/PyImageCourse_beginner/raw/mas
```

In [446…
```
image_rgb.shape
```

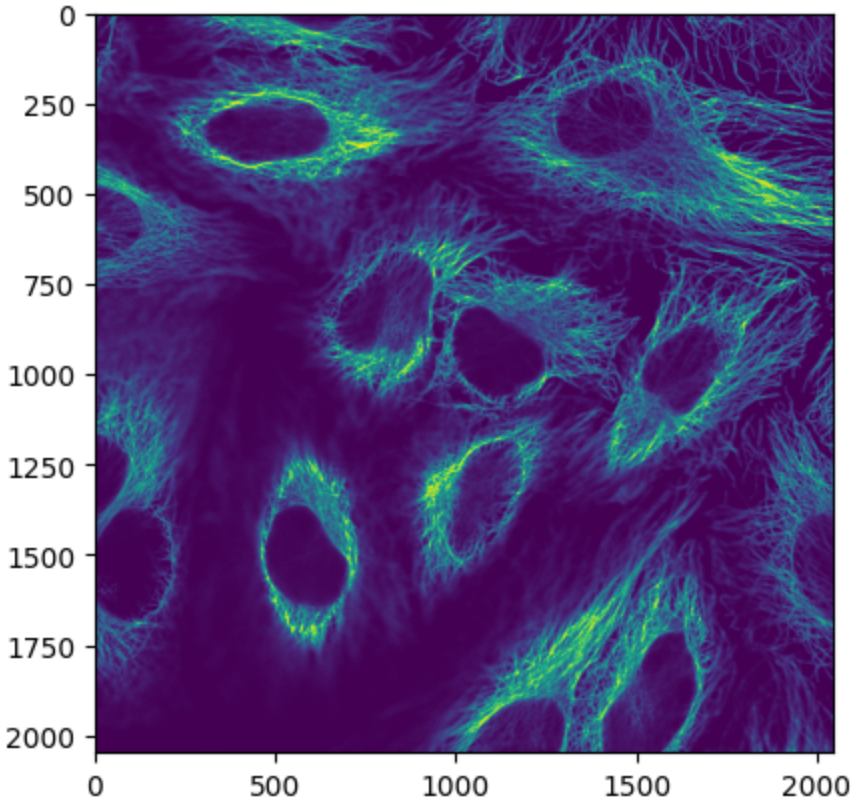
Out[446]:
```
(2048, 2048, 3)
```

In [447…
```
fig, ax = plt.subplots(figsize=(10,10))
ax.imshow(image_rgb);
```



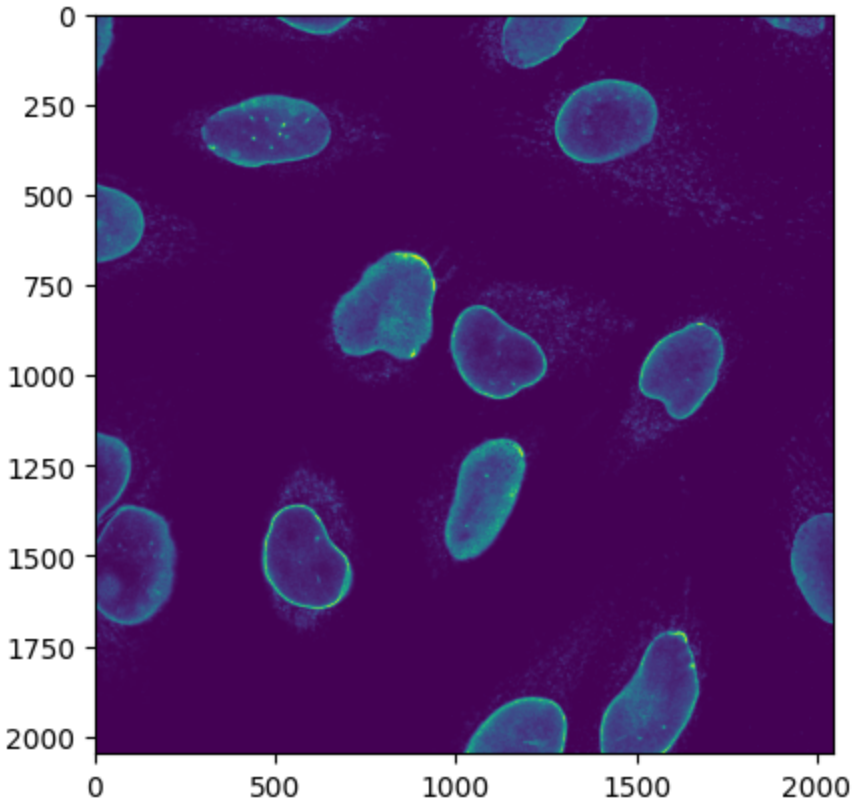
In [448…]
```
image_channel1 = image_rgb[:, : , 0]
```

In [449…]
```
plt.imshow(image_channel1);
```



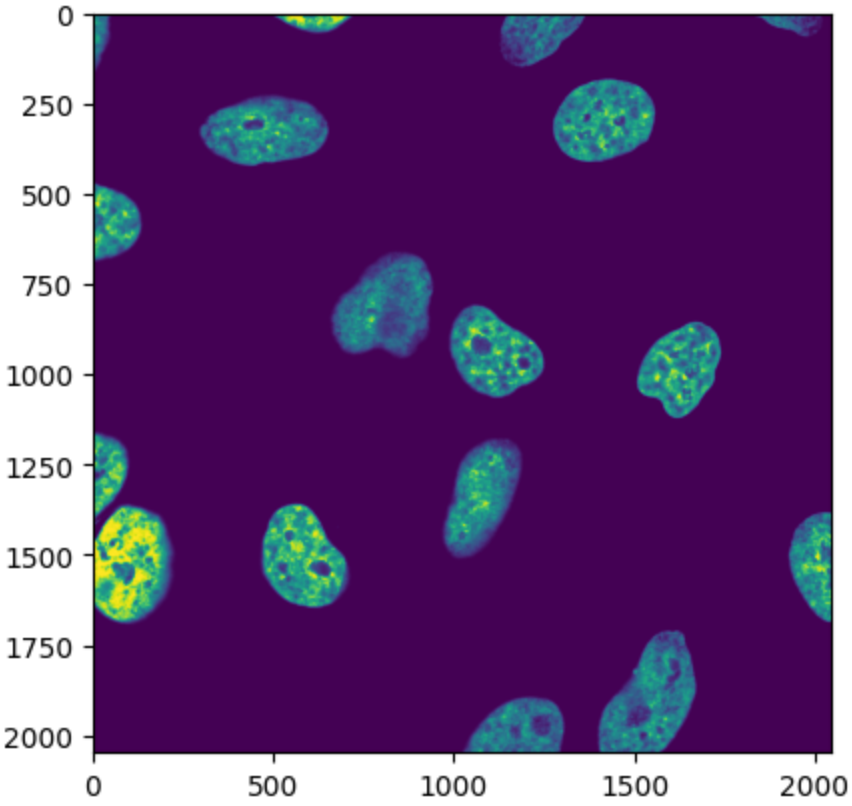
In [450…

```
image_channel2 = image_rgb[:, : , 1]
plt.imshow(image_channel2);
```



In [451…

```
image_channel3 = image_rgb[:, : , 2]
plt.imshow(image_channel3);
```



In [452…
```
image_channel1.shape
```

Out[452]:
```
(2048, 2048)
```

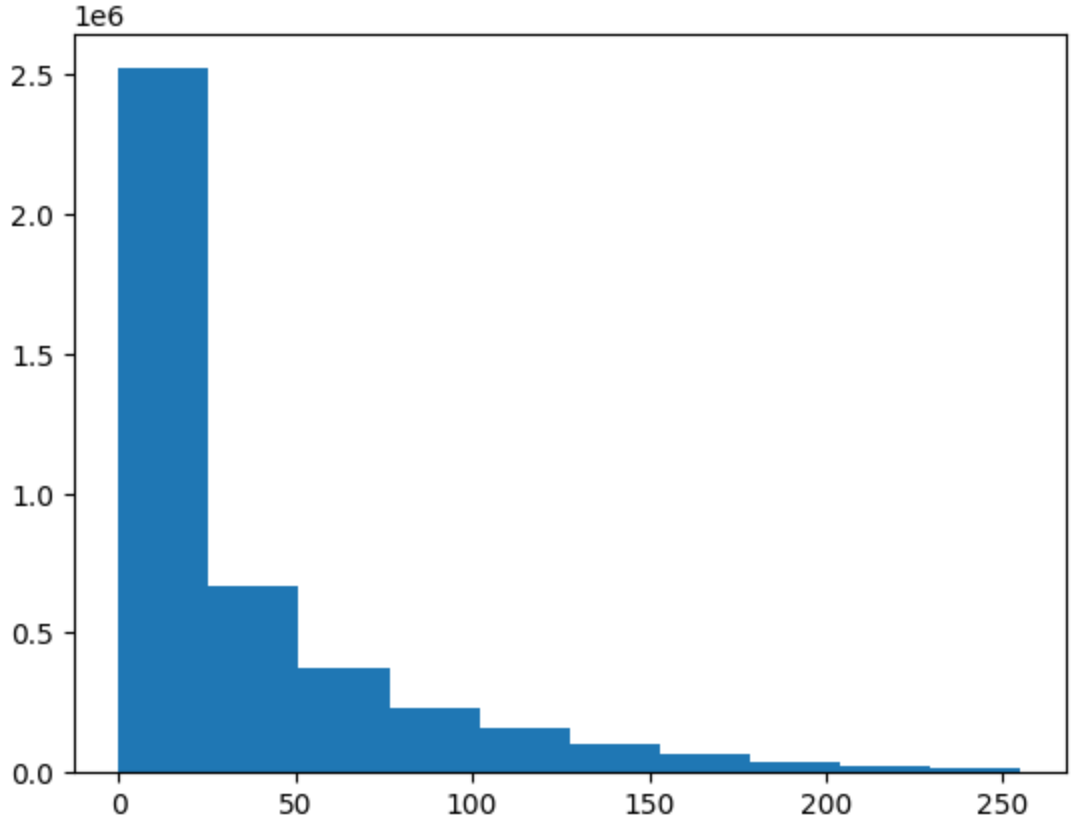
In [453…
```
2048 * 2048
```

Out[453]:
```
4194304
```

In [454…
```
image_channel1.ravel().shape
```

Out[454]:
```
(4194304,)
```

In [455…
```
plt.hist(image_channel1.ravel());
```



In [456…
```
image_channel1.max()
```

Out[456]:
```
255
```

In [457…
```
image_channel1.dtype
```

Out[457]:
```
dtype('uint8')
```

In [458…
```
image_channel1.dtype
```

Out[458]:
```
dtype('uint8')
```

In [459…
```
image_channel1[0:3, 0:3]
```

Out[459]:
```
array([[13, 15, 22],
       [22, 29, 36],
       [14, 22, 25]], dtype=uint8)
```

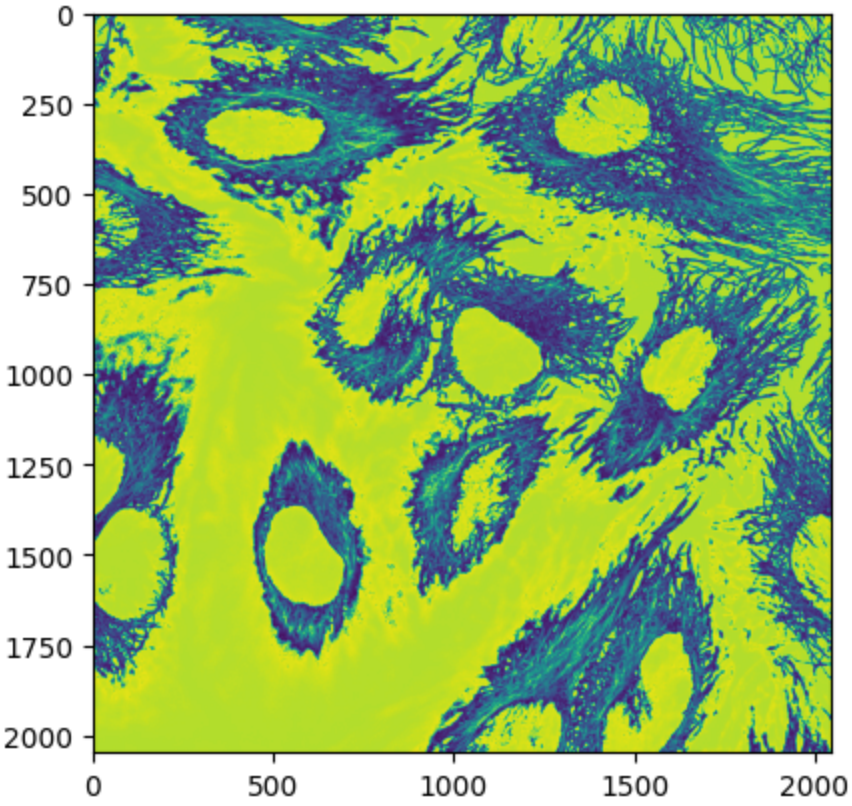
In [460…
```
image_channel1[0:3, 0:3] - 10
```

Out[460]:
```
array([[ 3,  5, 12],
       [12, 19, 26],
       [ 4, 12, 15]], dtype=uint8)
```

In [461…
```
image_channel1[0:3, 0:3] - 30
```

Out[461]:
```
array([[239, 241, 248],
       [248, 255,   6],
       [240, 248, 251]], dtype=uint8)
```

In [462…
```
plt.imshow(image_channel1-30);
```



In [463…
```
import numpy as np

image_channel1_float = image_channel1.astype(np.float16)
```

In [464…
```
image_channel1_float[0:3, 0:3]
```

Out[464]:
```
array([[13., 15., 22.],
       [22., 29., 36.],
       [14., 22., 25.]], dtype=float16)
```

In [465…
```
image_channel1_float[0:3, 0:3] - 30
```

Out[465]:
```
array([[-17., -15.,  -8.],
       [ -8.,  -1.,   6.],
       [-16.,  -8.,  -5.]], dtype=float16)
```

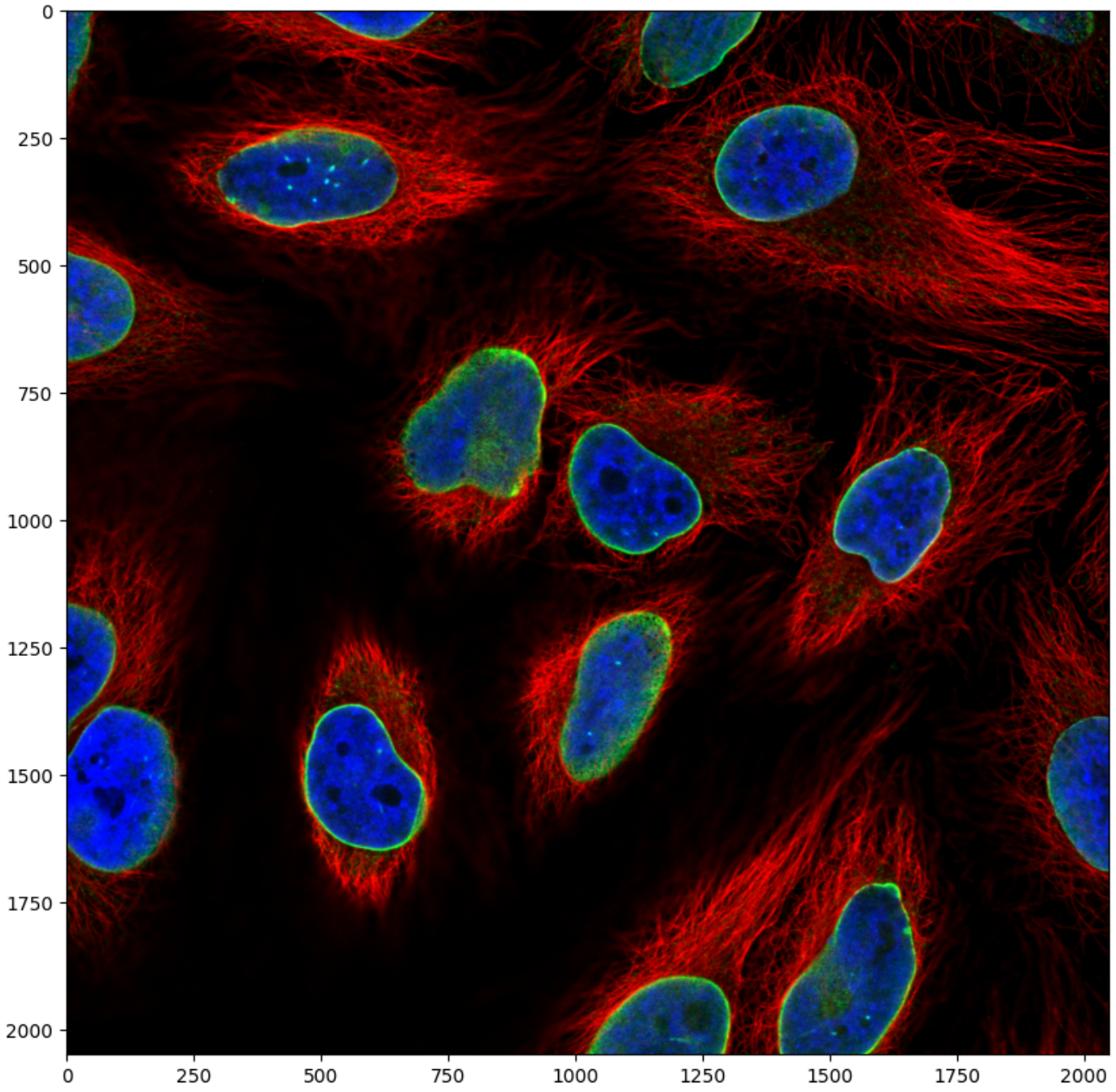
In [466…
```
import numpy as np
import matplotlib.pyplot as plt
import skimage
import skimage.io
```

In [467…
```
#image_stack = skimage.io.imread('images/46658_784_B12_1.tif')
image_stack = skimage.io.imread('https://github.com/guiwitz/PyImageCourse_beginner/raw/m
```

In [468…
```
image_stack.shape
```

Out[468]:
```
(2048, 2048, 3)
```

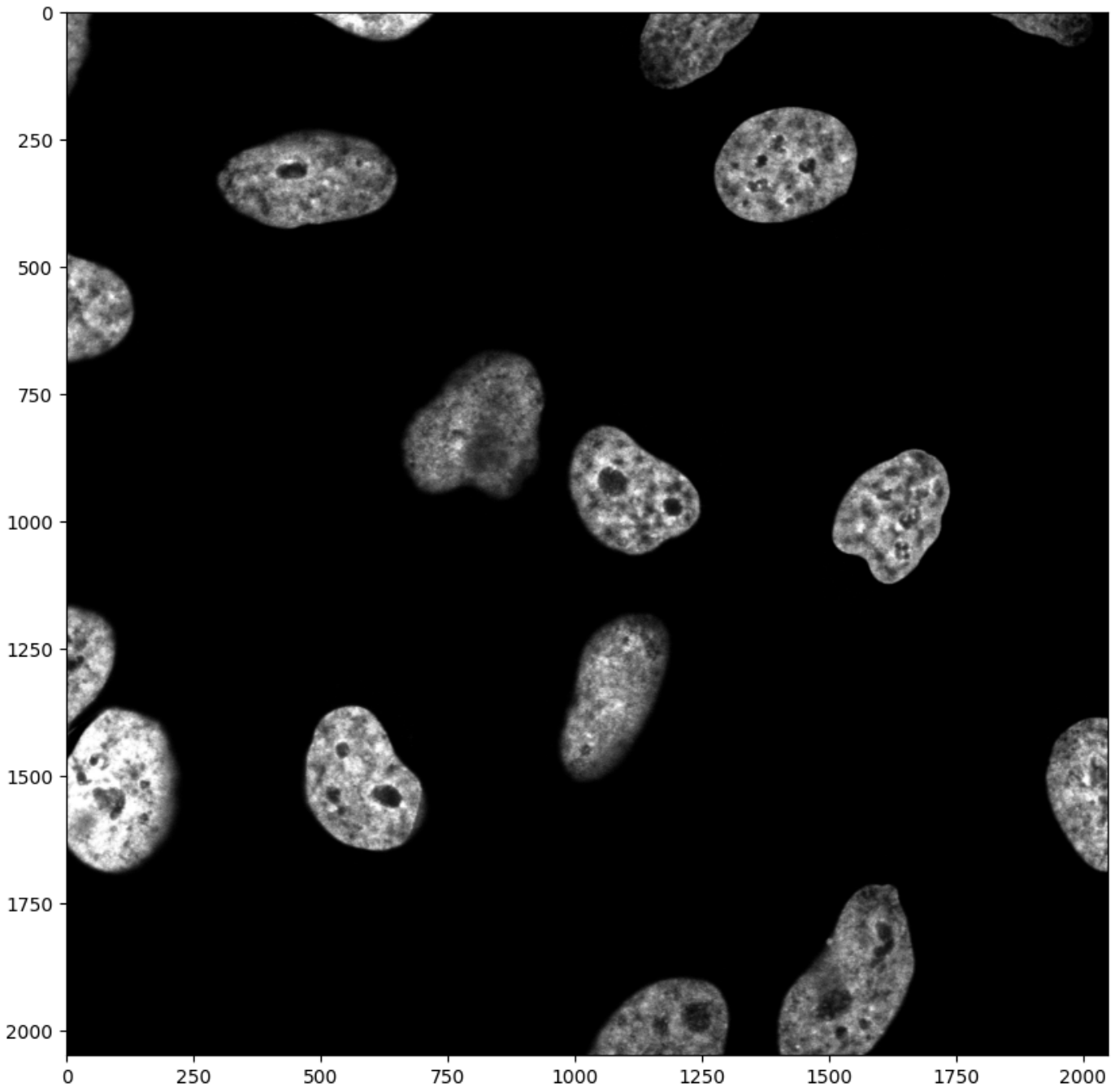
In [469…
```
plt.subplots(figsize=(10,10))
plt.imshow(image_stack);
```



In [470…
```
image_nuclei = image_stack[:,:,2]
```

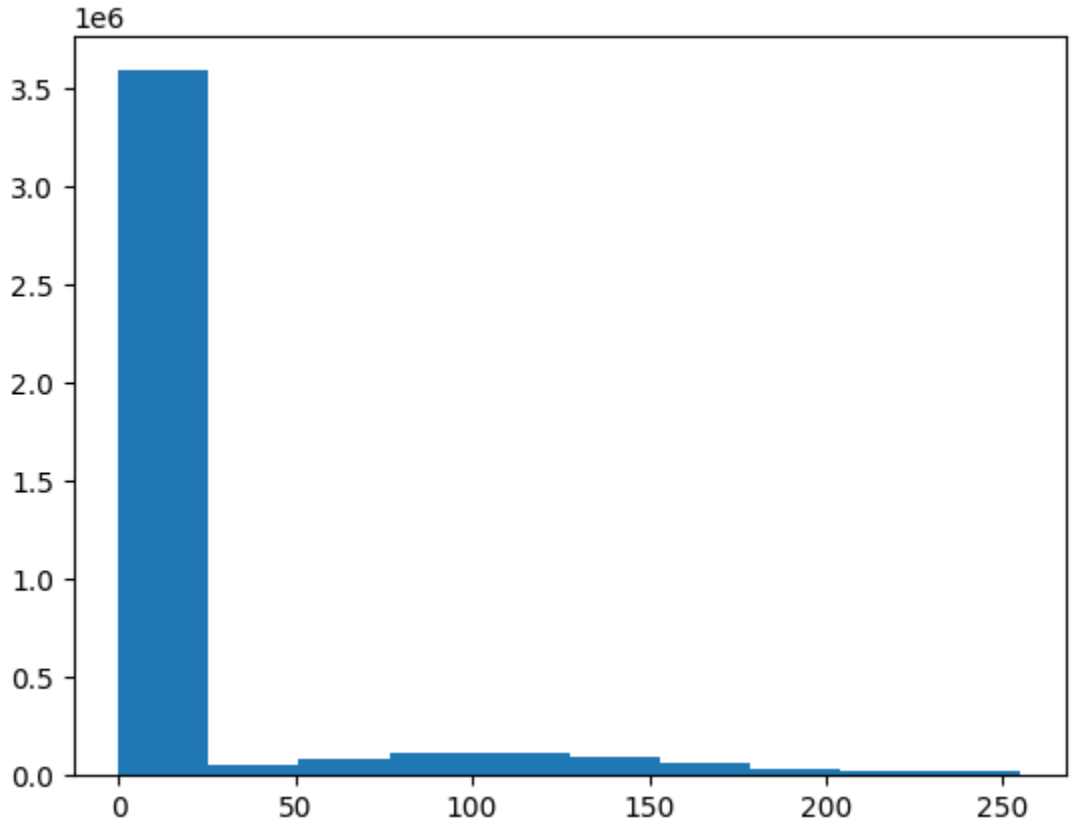
In [471…
```
image_nuclei = image_stack[:,:,2]
```

In [472…
```
plt.subplots(figsize=(10,10))
plt.imshow(image_nuclei, cmap = 'gray');
```

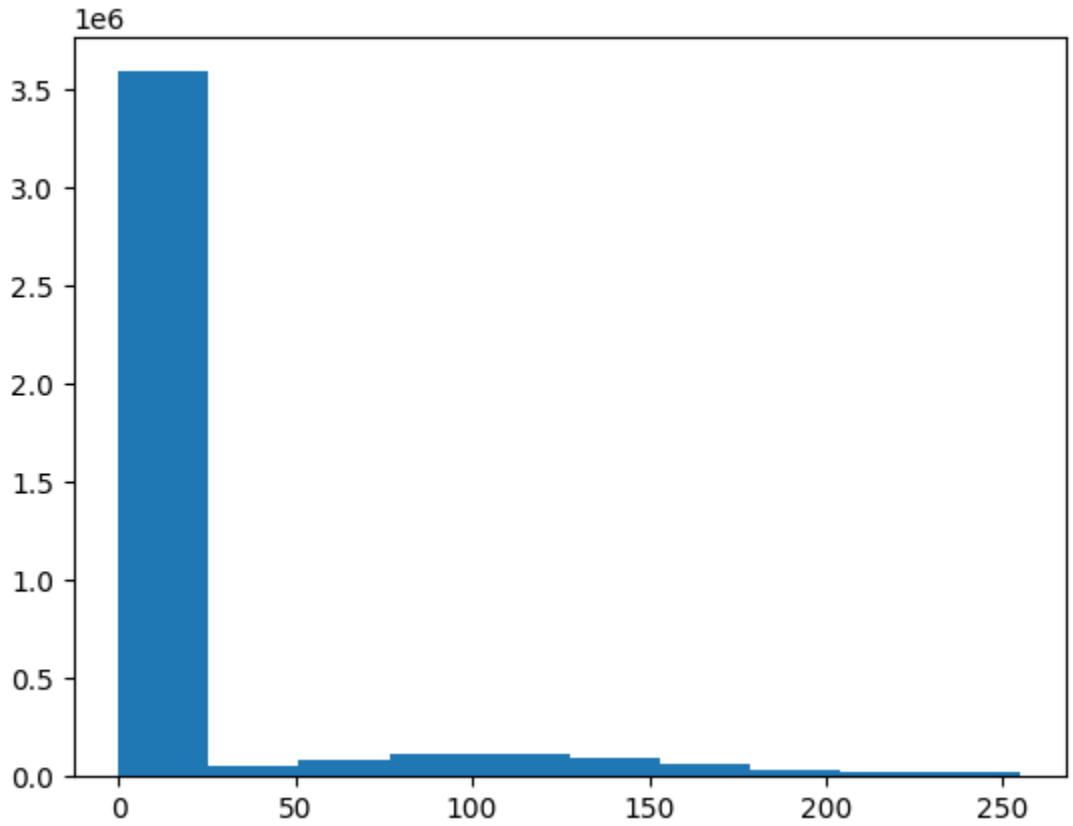


In [473…

```
plt.hist(np.ravel(image_nuclei));
```

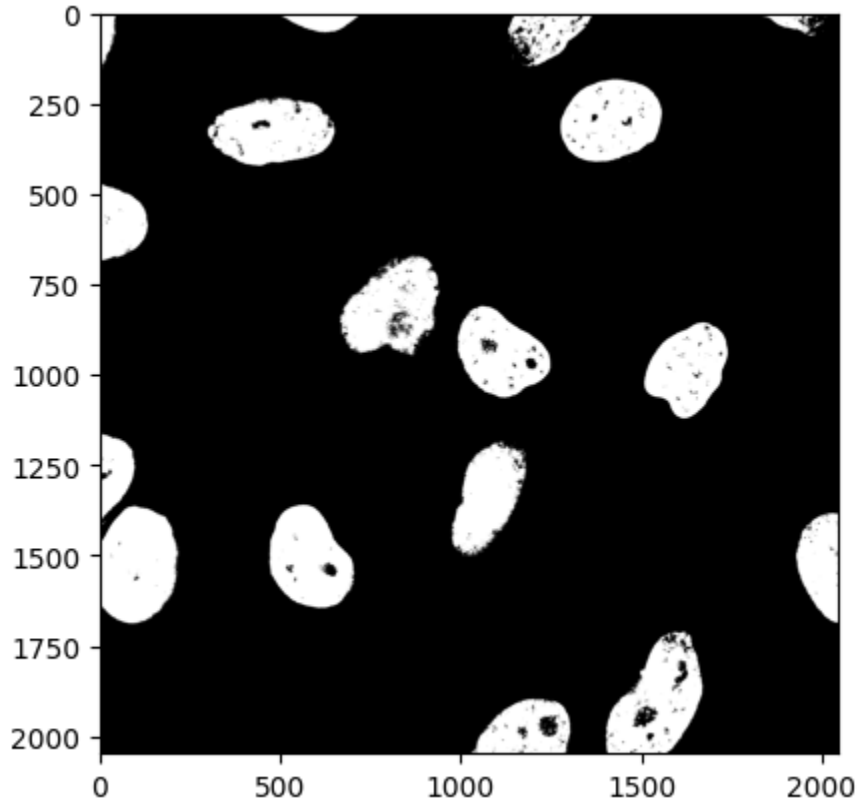


In [474…
```
plt.hist(np.ravel(image_nuclei));
```



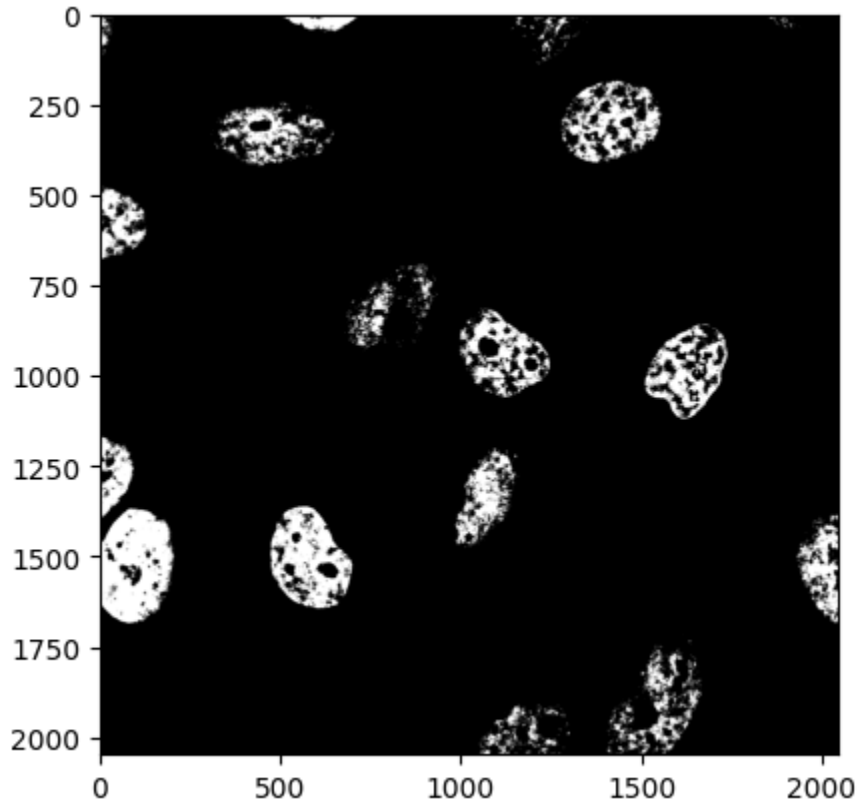
In [475…
```
mask = image_nuclei > 50
```

In [476…
```
plt.imshow(mask, cmap = 'gray');
```



In [477…

```
mask = image_nuclei > 120
plt.imshow(mask, cmap = 'gray');
```



In [478…

```
import skimage.filters

help(skimage.filters.threshold_otsu)
```

```
Help on function threshold_otsu in module skimage.filters.thresholding:

threshold_otsu(image=None, nbins=256, *, hist=None)
    Return threshold value based on Otsu's method.
```

```
    Either image or hist must be provided. If hist is provided, the actual
    histogram of the image is ignored.

    Parameters
    ----------
    image : (N, M[, ..., P]) ndarray, optional
        Grayscale input image.
    nbins : int, optional
        Number of bins used to calculate histogram. This value is ignored for
        integer arrays.
    hist : array, or 2-tuple of arrays, optional
        Histogram from which to determine the threshold, and optionally a
        corresponding array of bin center intensities. If no hist provided,
        this function will compute it from the image.


    Returns
    -------
    threshold : float
        Upper threshold value. All pixels with an intensity higher than
        this value are assumed to be foreground.

    References
    ----------
    .. [1] Wikipedia, https://en.wikipedia.org/wiki/Otsu's_Method

    Examples
    --------
    >>> from skimage.data import camera
    >>> image = camera()
    >>> thresh = threshold_otsu(image)
    >>> binary = image <= thresh

    Notes
    -----
    The input image must be grayscale.
```

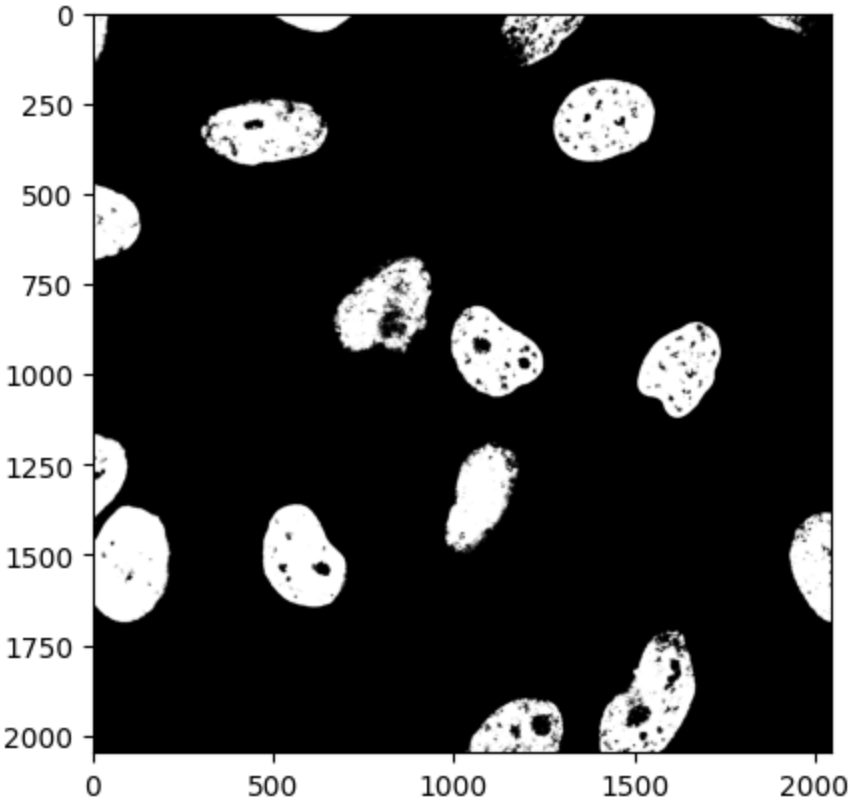
In [479…
```
my_otsu_threshold = skimage.filters.threshold_otsu(image_nuclei)
print(my_otsu_threshold)
```

```
66
```

In [480…
```
mask_nuclei = image_nuclei > my_otsu_threshold
```

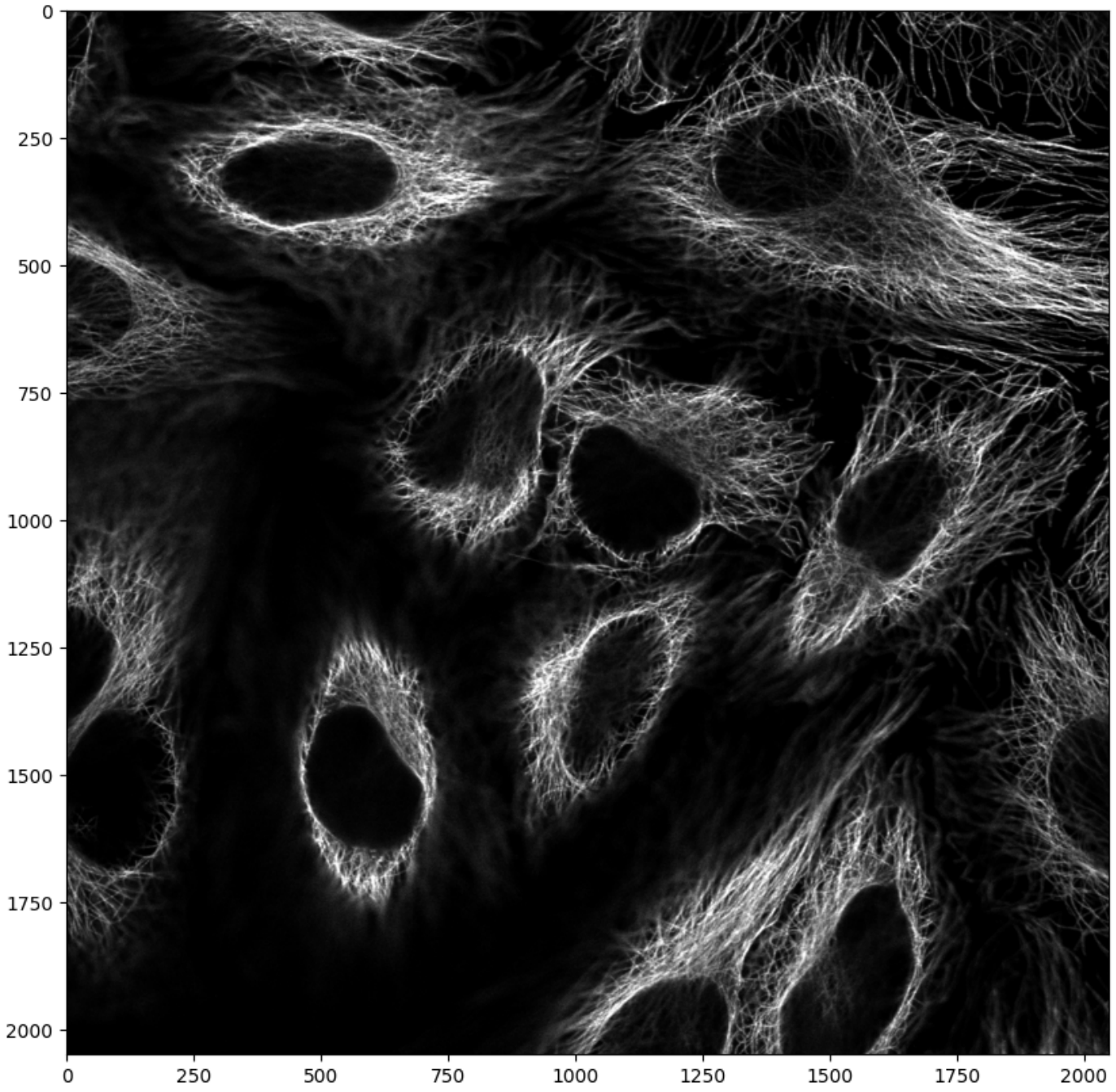
In [481…
```
mask_nuclei = image_nuclei > my_otsu_threshold
```

In [482…
```
plt.imshow(mask_nuclei, cmap = 'gray');
```



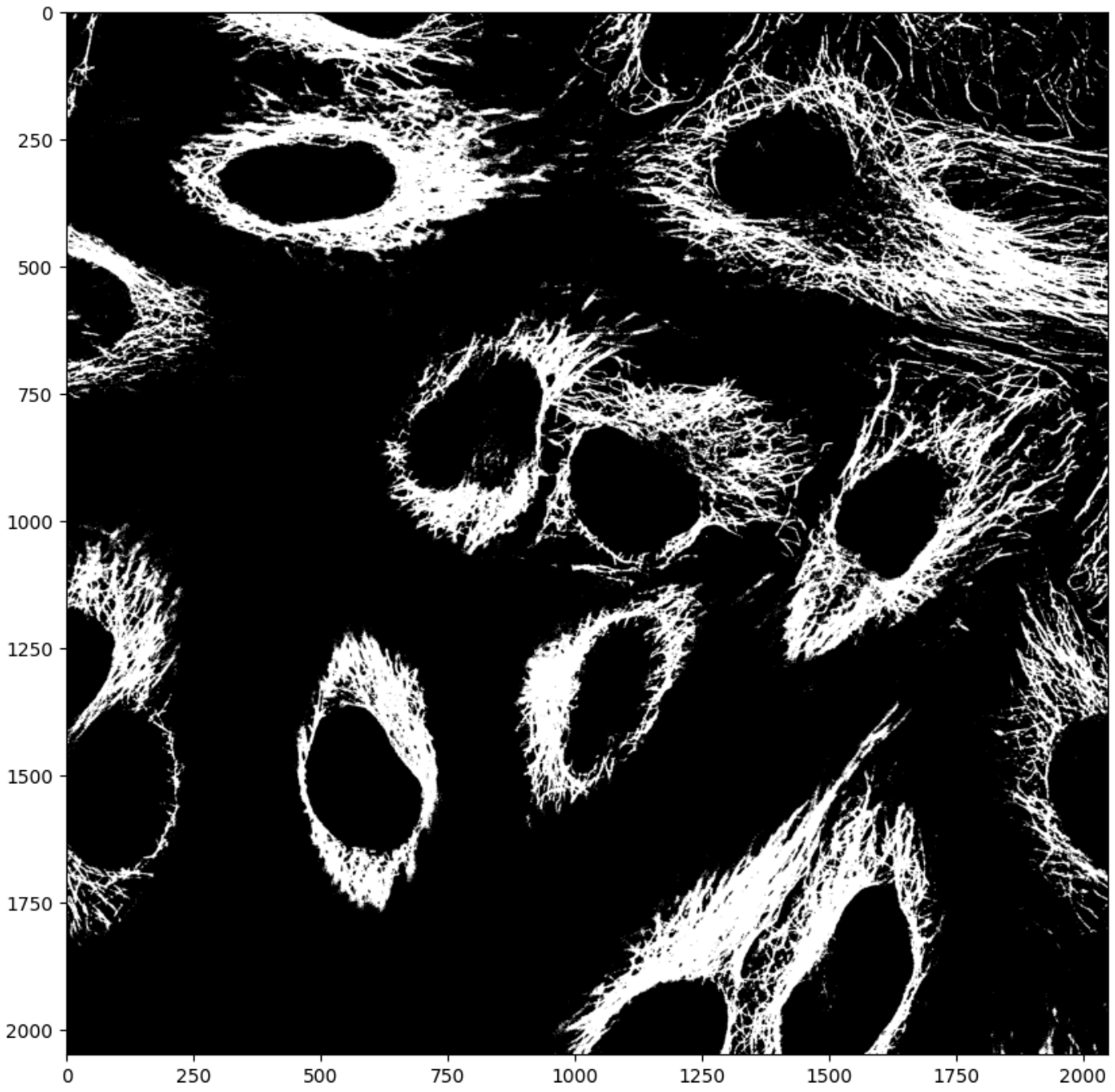
In [483…
```
image_cells = image_stack[:,:,0]
```

In [484…
```
plt.subplots(figsize=(10,10))
plt.imshow(image_cells, cmap = 'gray');
```



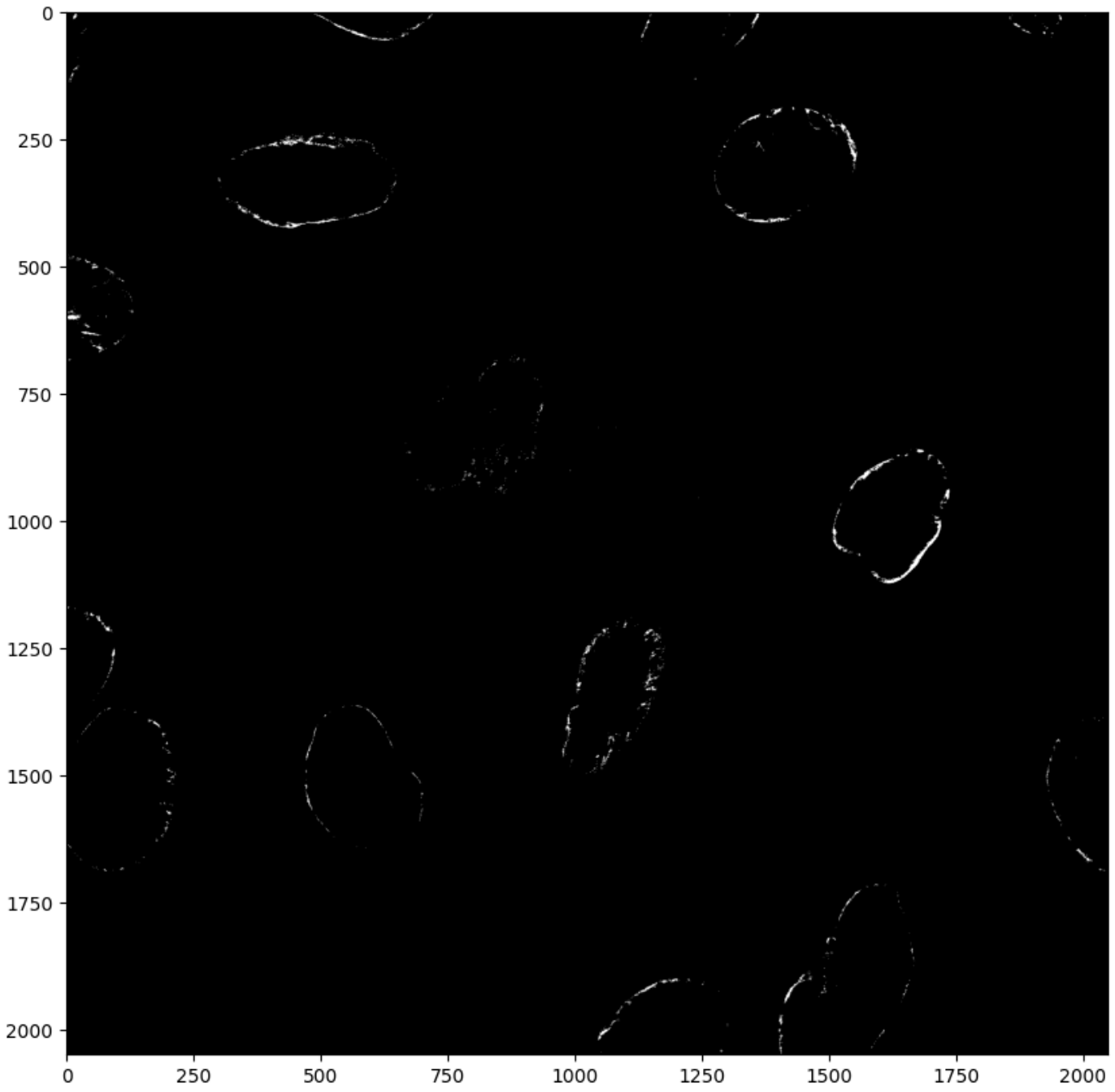
In [485…
```
mask_cells = image_cells > skimage.filters.threshold_otsu(image_cells)
```

In [486…
```
plt.subplots(figsize=(10,10))
plt.imshow(mask_cells, cmap = 'gray');
```



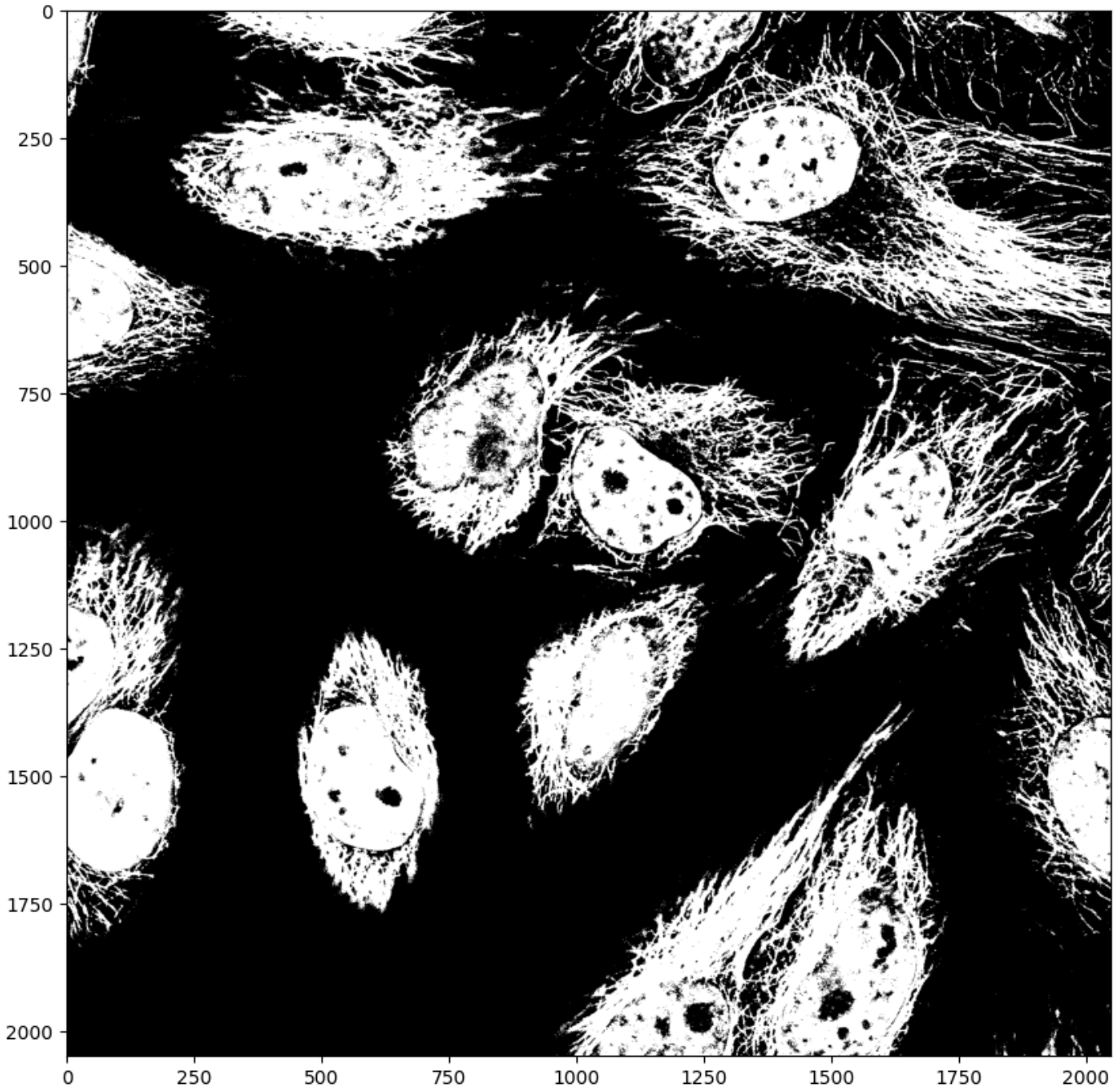
In [487…
```
both_masks = mask_cells * mask_nuclei
```

In [488…
```
plt.subplots(figsize=(10,10))
plt.imshow(both_masks, cmap = 'gray');
```



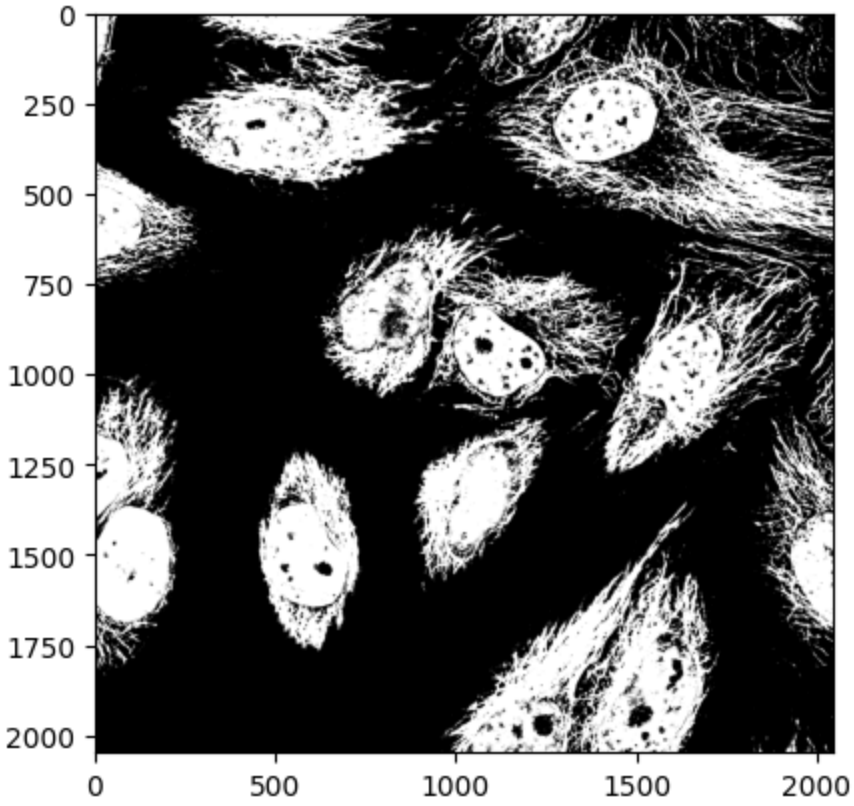
In [489…

```
plt.subplots(figsize=(10,10))
plt.imshow(mask_cells + mask_nuclei, cmap = 'gray');
```



In [490…

```
combine_or = np.logical_or(mask_cells, mask_nuclei)
plt.imshow(combine_or, cmap = 'gray');
```



In [491…

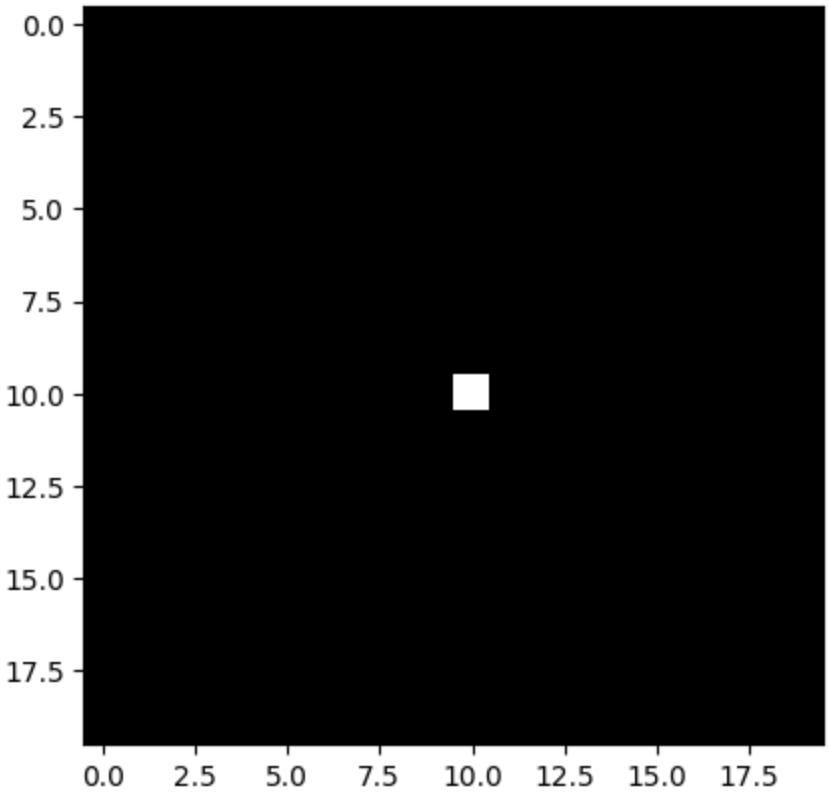
```
# from IPython.display import HTML, display
# HTML(filename='https://cildata.crbs.ucsd.edu/media/images/13901/13901.tif')
```

In [492…

```
import numpy as np
import matplotlib.pyplot as plt
import skimage
import skimage.io
import skimage.morphology
```

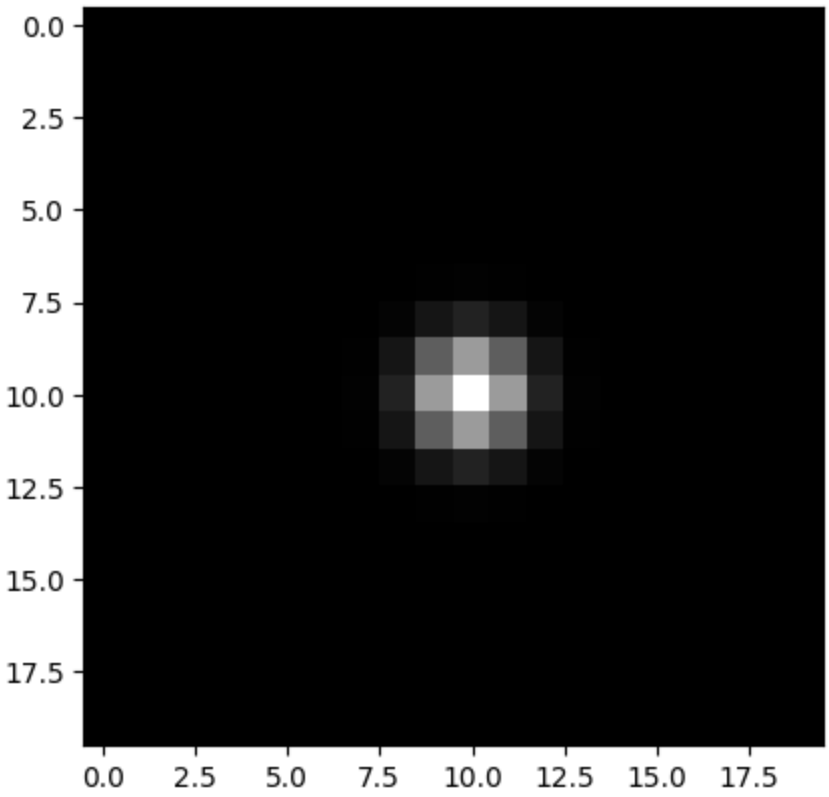
In [493…

```
single_dot = np.zeros((20,20))
single_dot[10,10] = 1
plt.imshow(single_dot, cmap = 'gray');
```

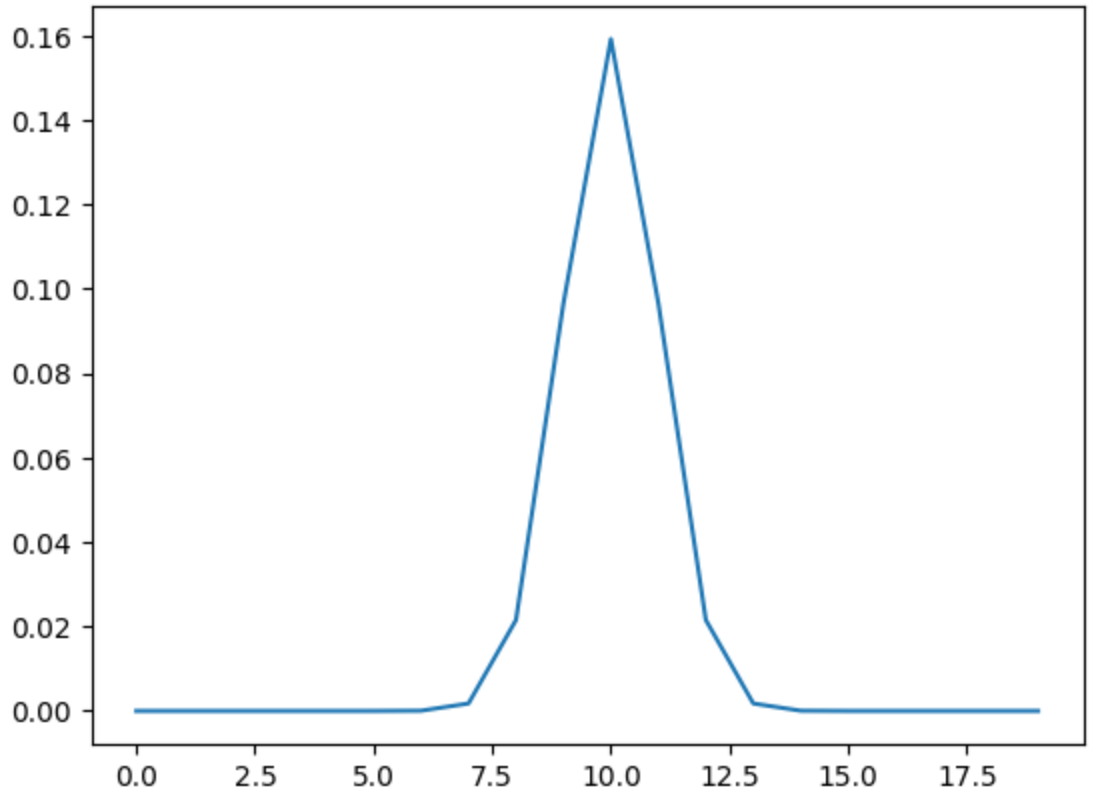


In [494…
```
filtered = skimage.filters.gaussian(single_dot)
```

In [495…
```
plt.imshow(filtered, cmap = 'gray');
```

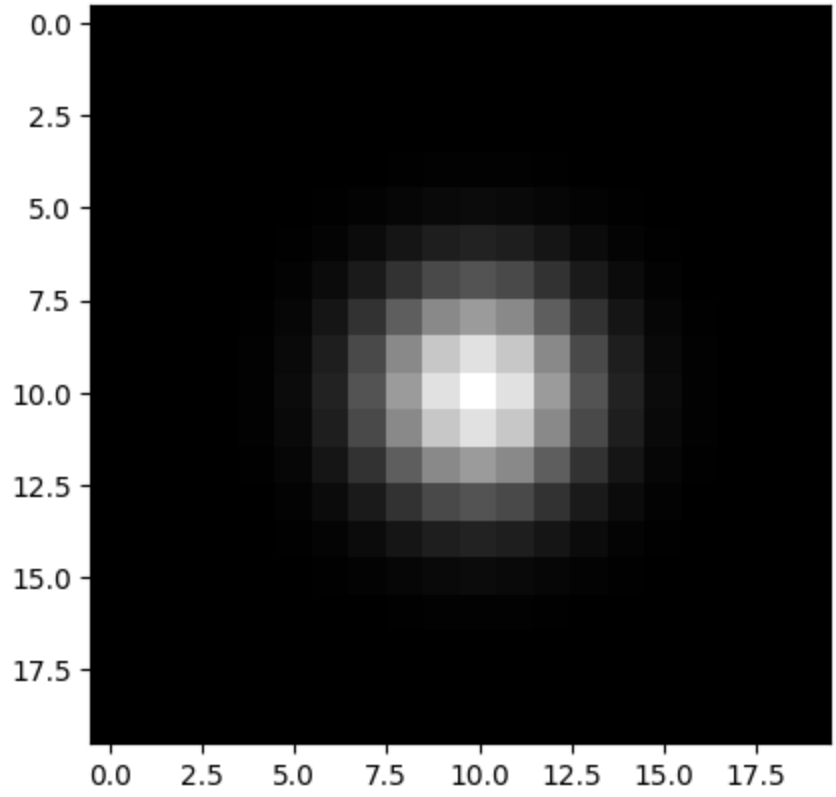


In [496…
```
plt.plot(filtered[10,:]);
```



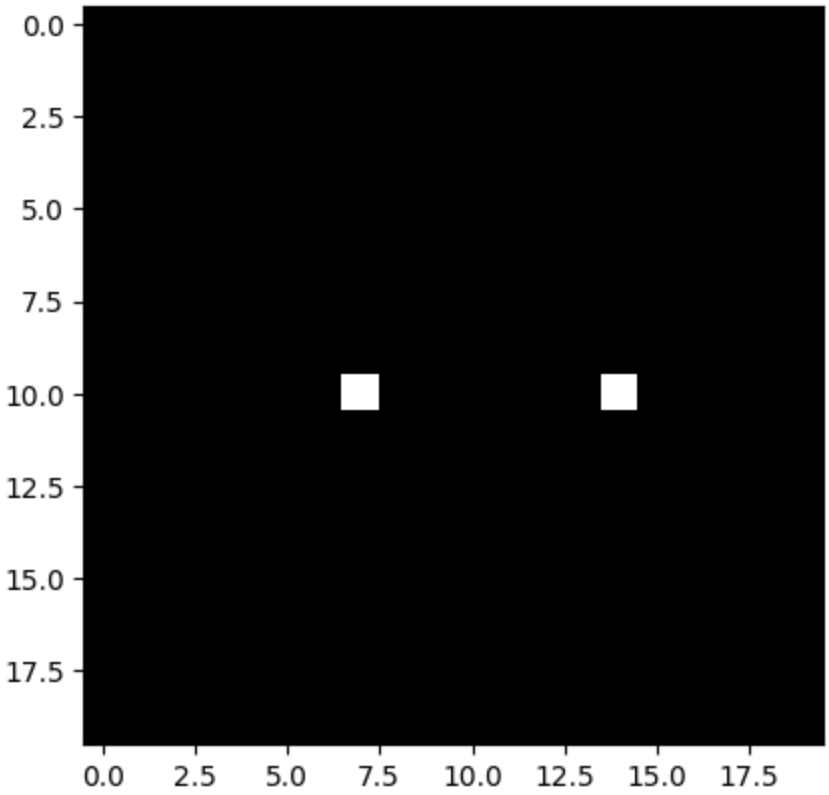
In [497…

```
filtered = skimage.filters.gaussian(single_dot,sigma=2)
plt.imshow(filtered, cmap = 'gray');
```



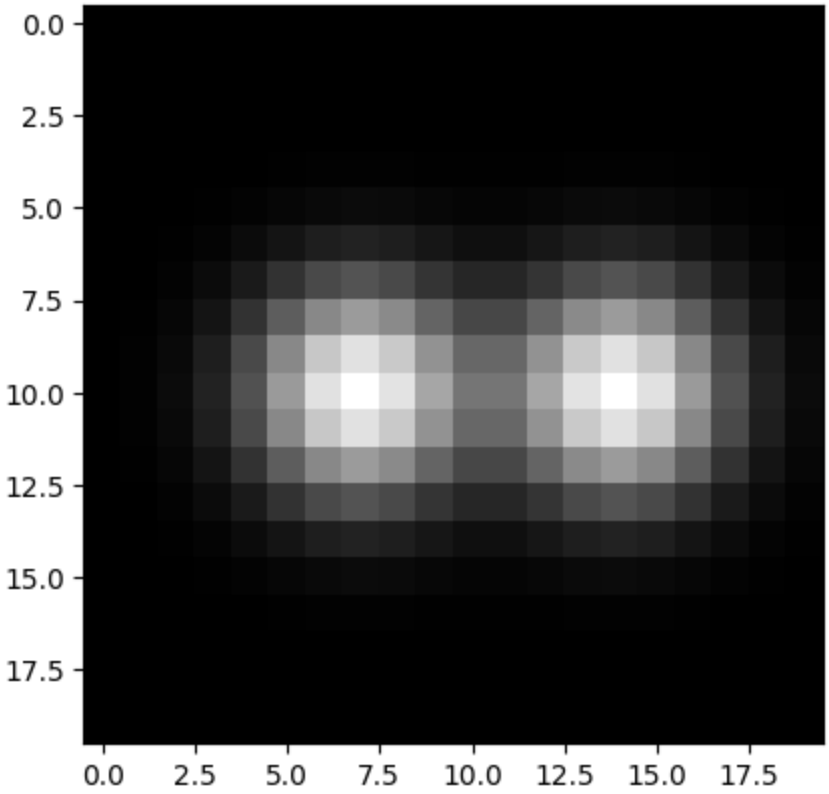
In [498…

```
double_dot = np.zeros((20,20))
double_dot[10,7] = 1
double_dot[10,14] = 1
plt.imshow(double_dot, cmap = 'gray');
```



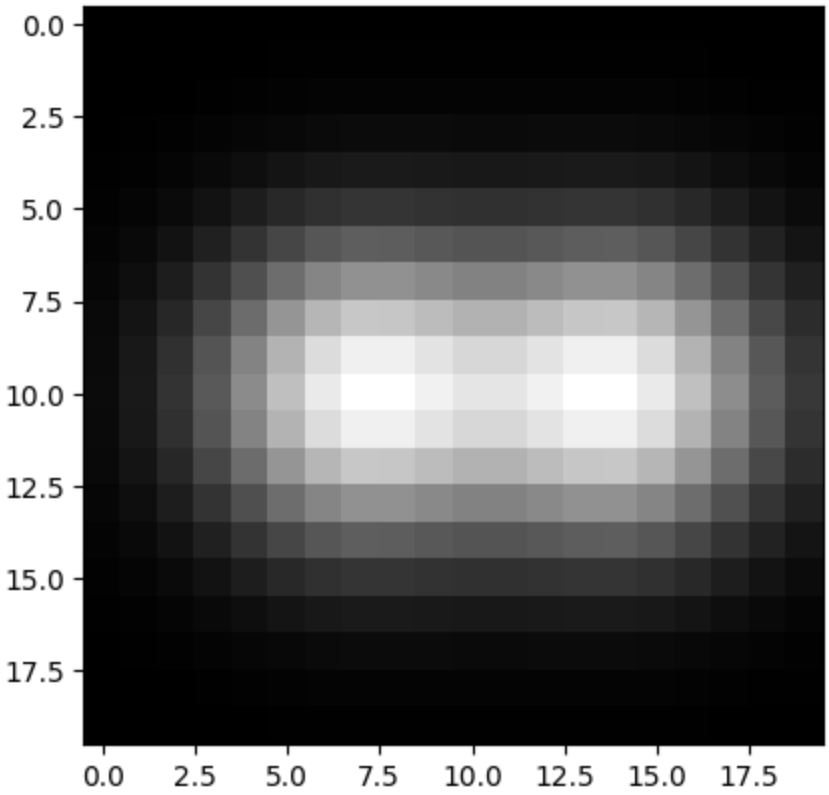
In [499…

```
filtered = skimage.filters.gaussian(double_dot,sigma=2)
plt.imshow(filtered, cmap = 'gray');
```



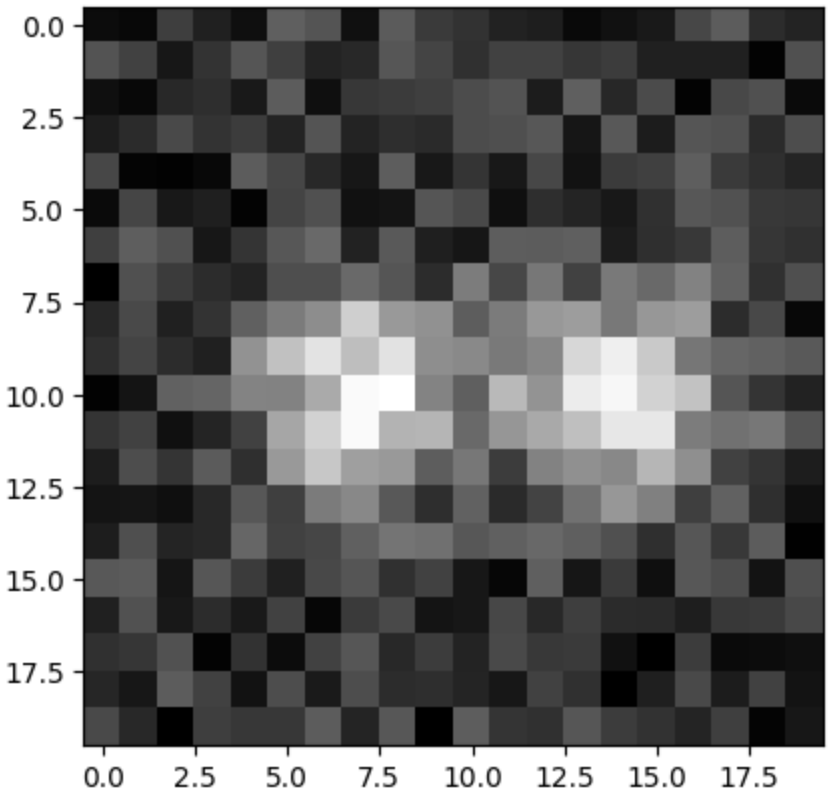
In [500…

```
filtered2 = skimage.filters.gaussian(filtered,sigma=2)
plt.imshow(filtered2, cmap = 'gray');
```

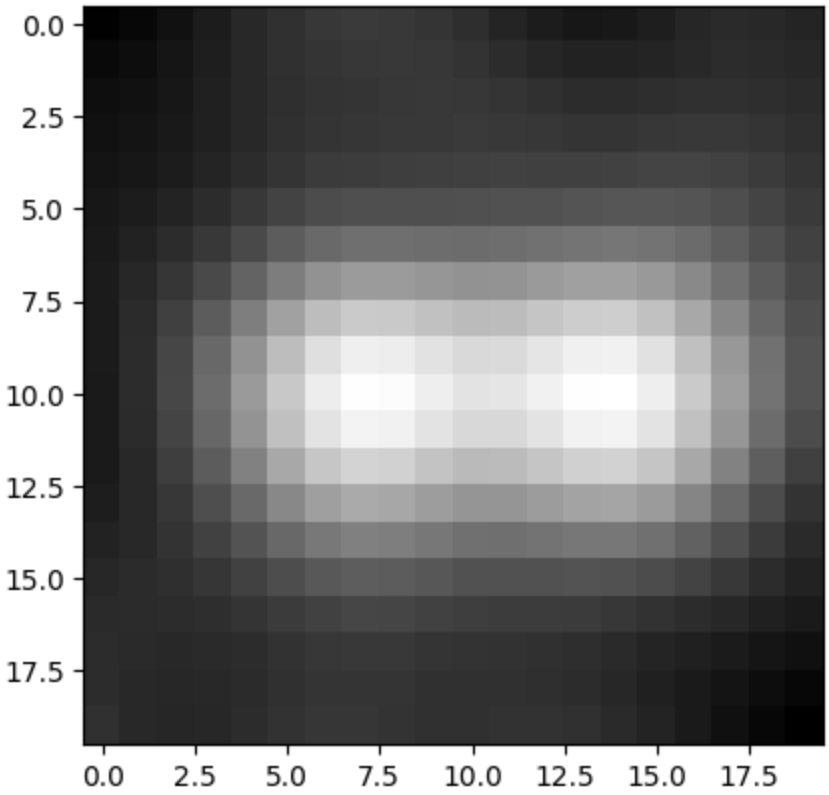


In [501…
```
noisy = filtered + 0.02*np.random.rand(20,20)
```

In [502…
```
plt.imshow(noisy, cmap = 'gray');
```

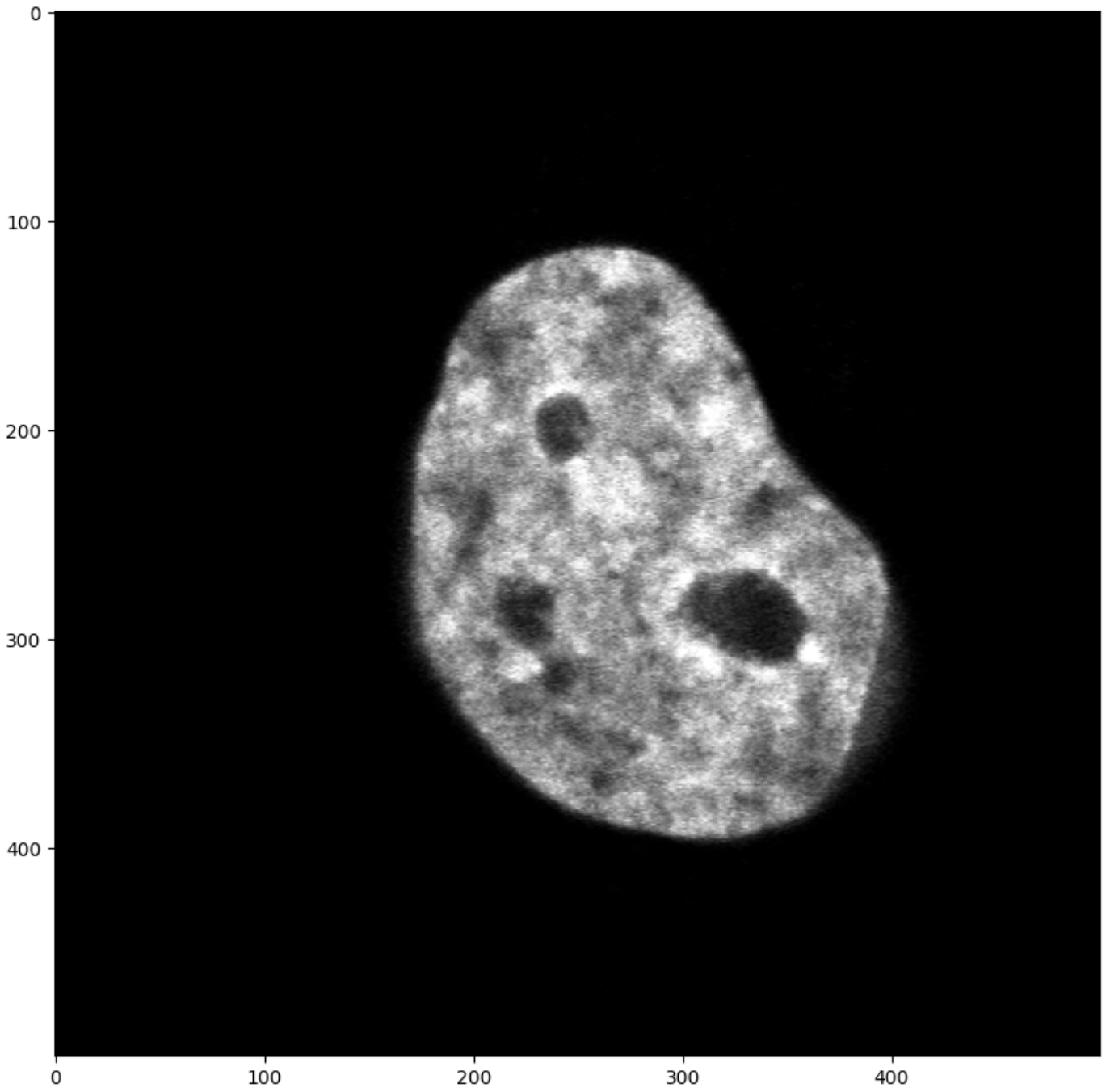


In [503…
```
noise_filtered = skimage.filters.gaussian(noisy,sigma=2)
plt.imshow(noise_filtered, cmap = 'gray');
```



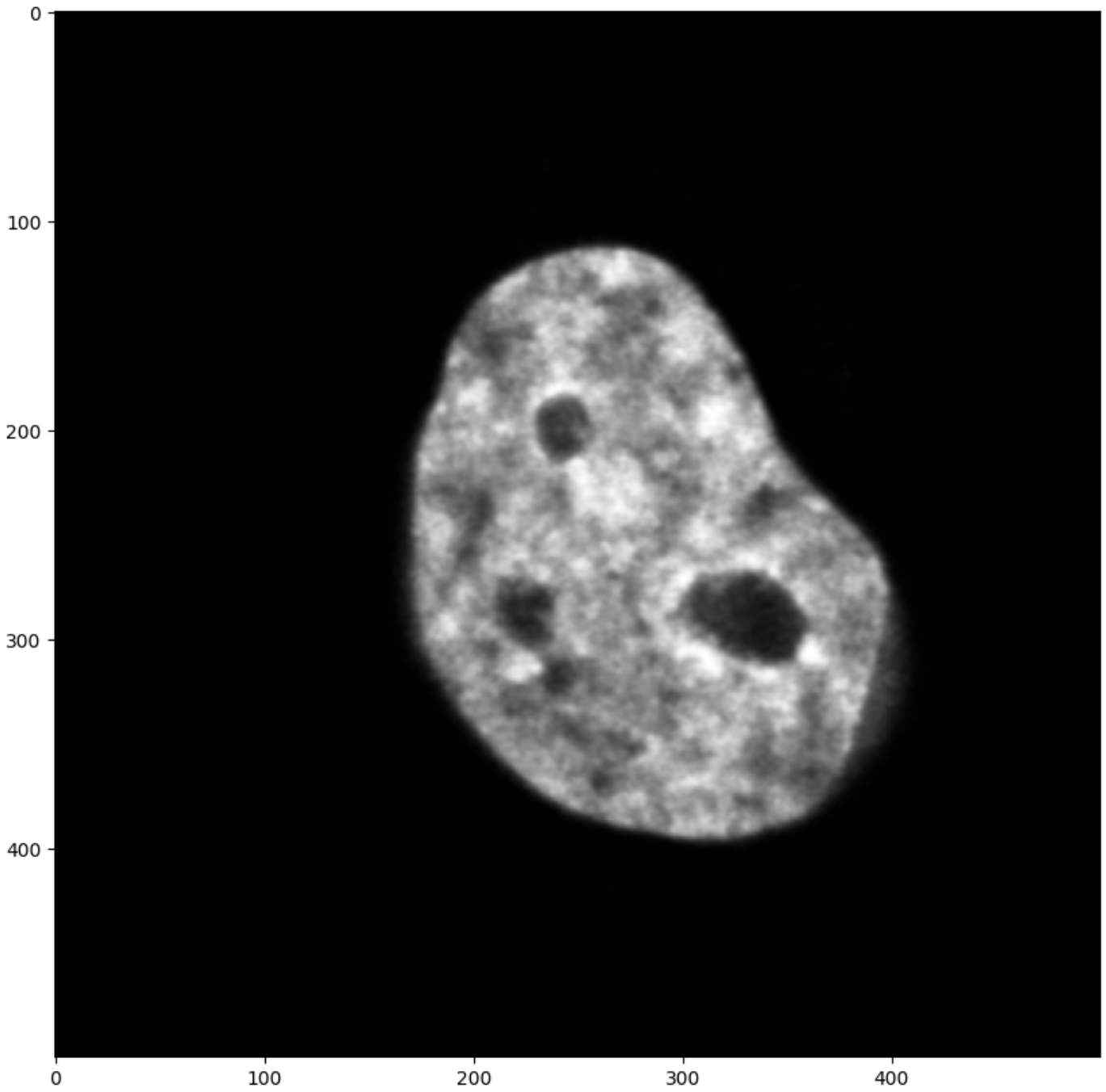
In [504…
```
#image_stack = skimage.io.imread('images/46658_784_B12_1.tif')
image_stack = skimage.io.imread('https://github.com/guiwitz/PyImageCourse_beginner/raw/m
image_nuclei = image_stack[1250:1750,300:800,2]
```

In [505…
```
plt.subplots(figsize=(10,10))
plt.imshow(image_nuclei, cmap = 'gray');
```



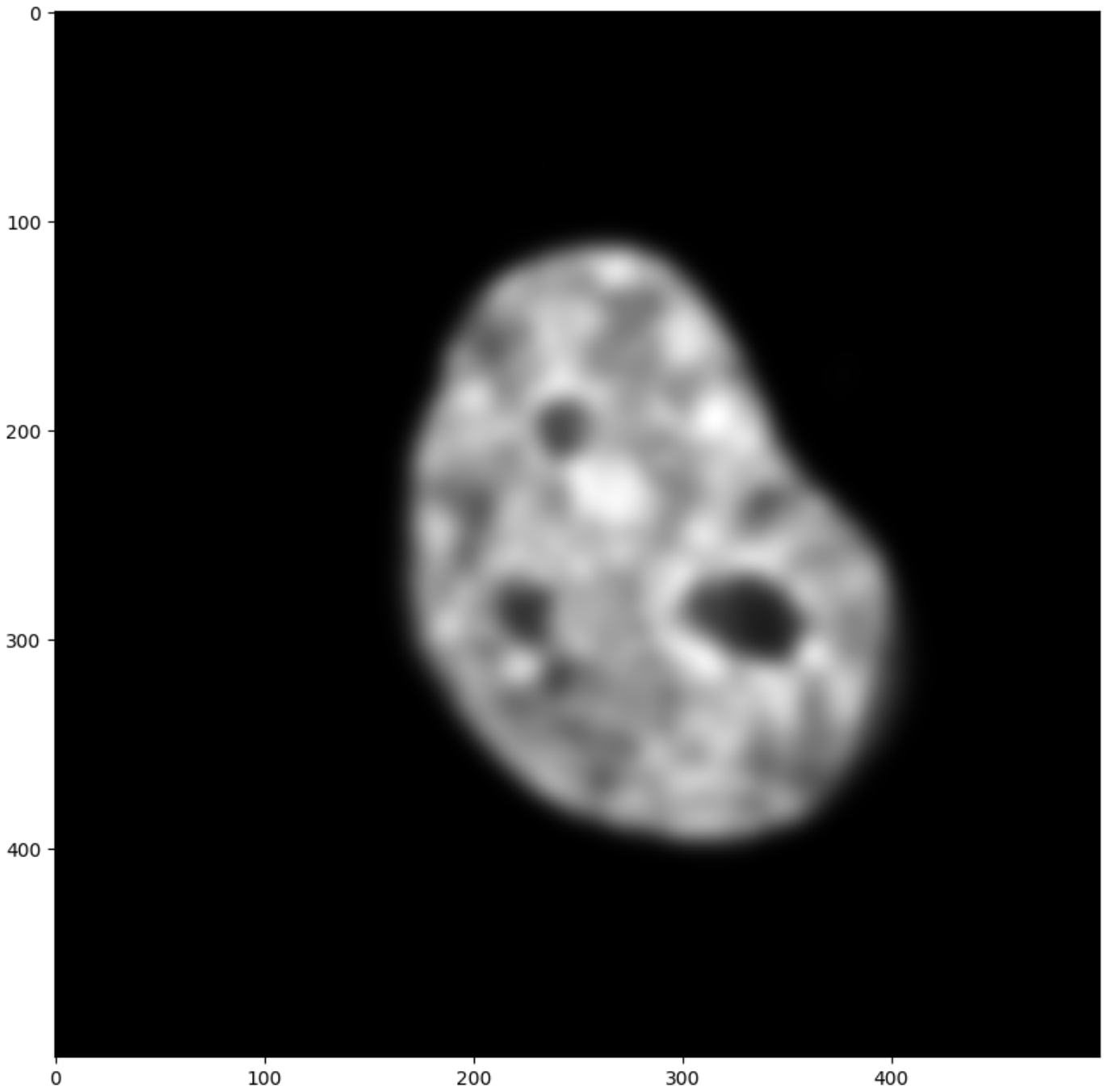
In [506…
```
filtered_image = skimage.filters.gaussian(image_nuclei)
```

In [507…
```
plt.subplots(figsize=(10,10))
plt.imshow(filtered_image, cmap = 'gray');
```



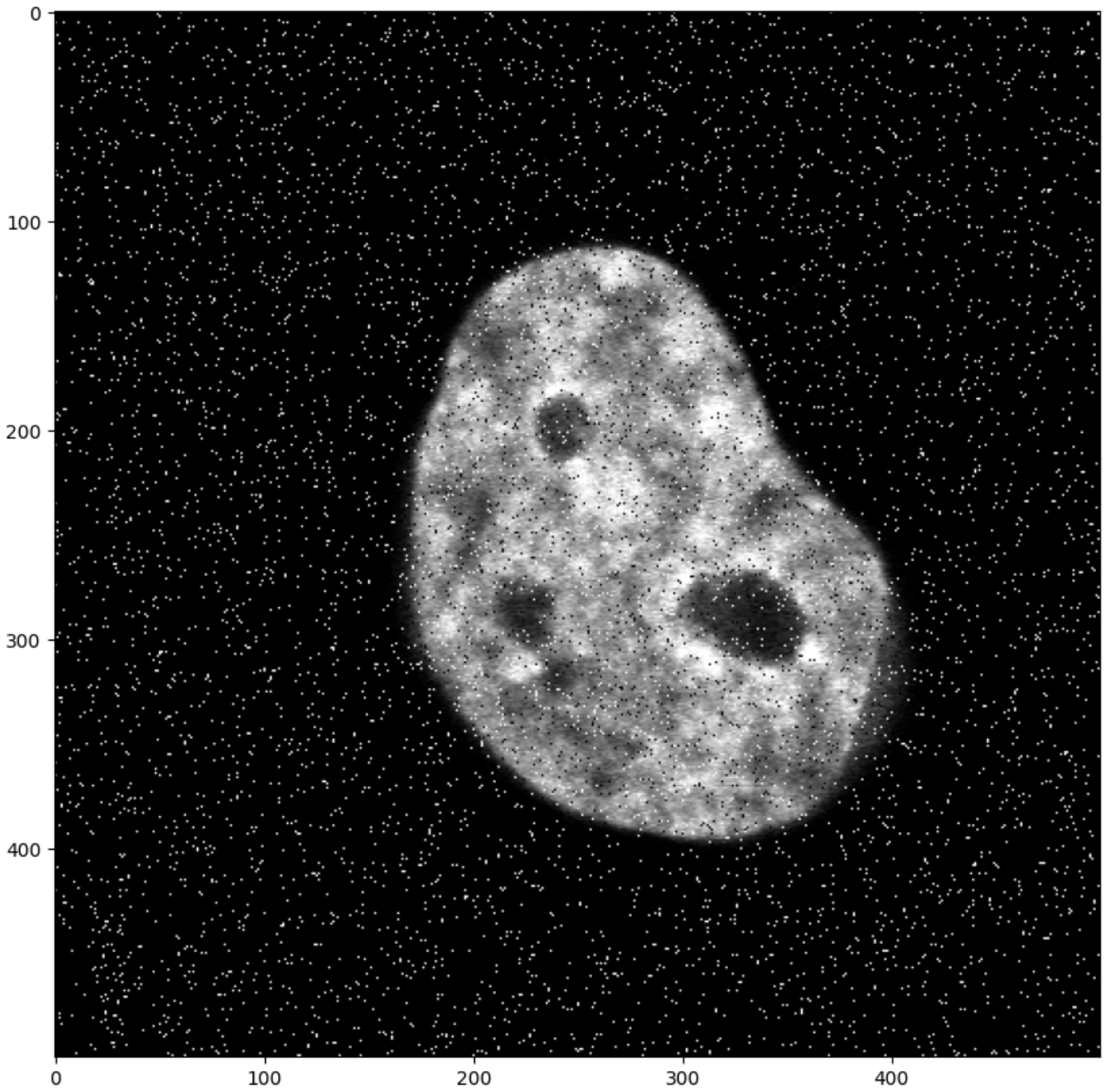
In [508…
```
filtered_image = skimage.filters.gaussian(image_nuclei, sigma = 5)
```

In [509…
```
plt.subplots(figsize=(10,10))
plt.imshow(filtered_image, cmap = 'gray');
```



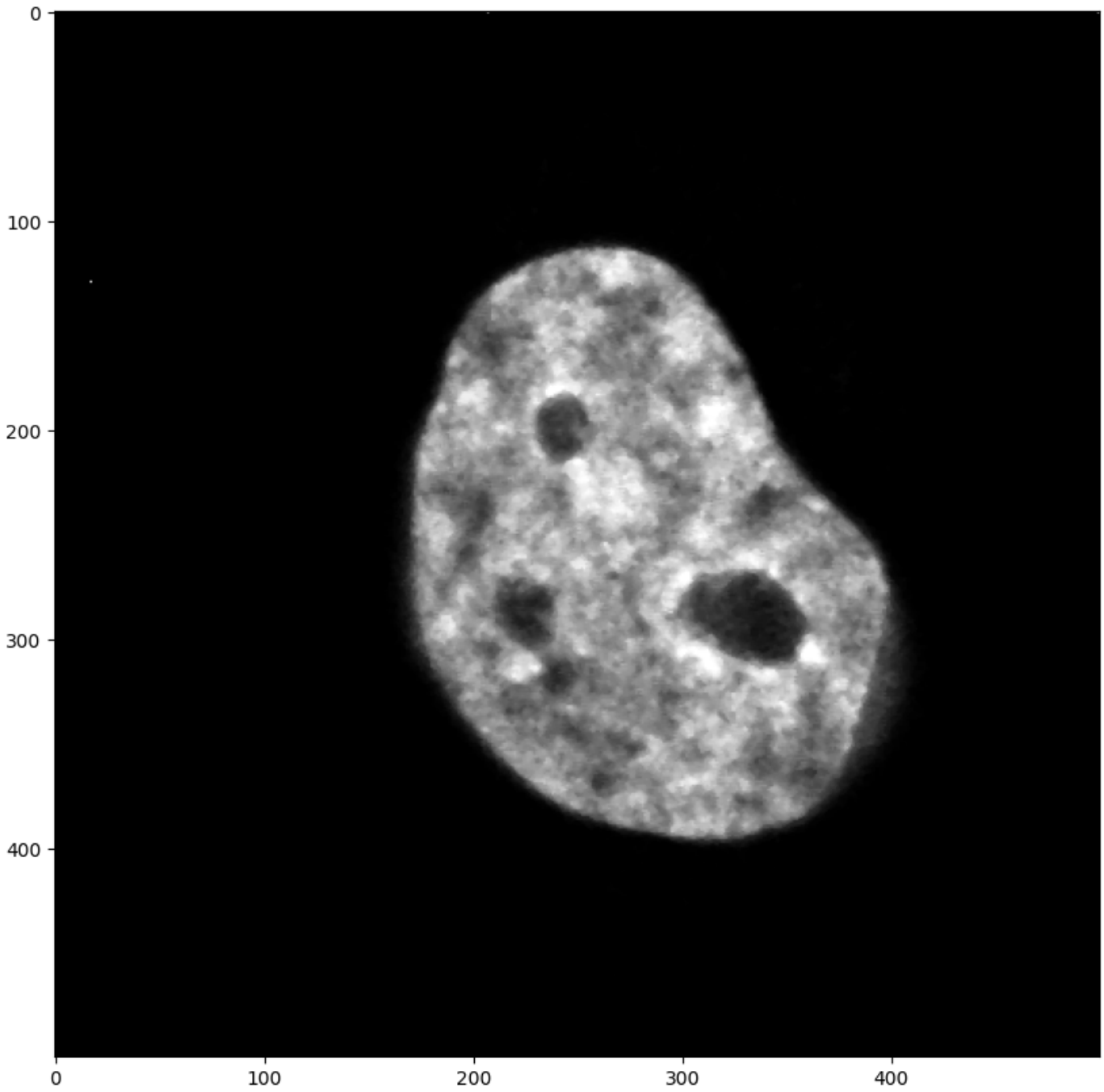
In [510…
```
image_noisy = skimage.util.random_noise(image_nuclei, mode='s&p')
```

In [511…
```
plt.subplots(figsize=(10,10))
plt.imshow(image_noisy, cmap = 'gray');
```



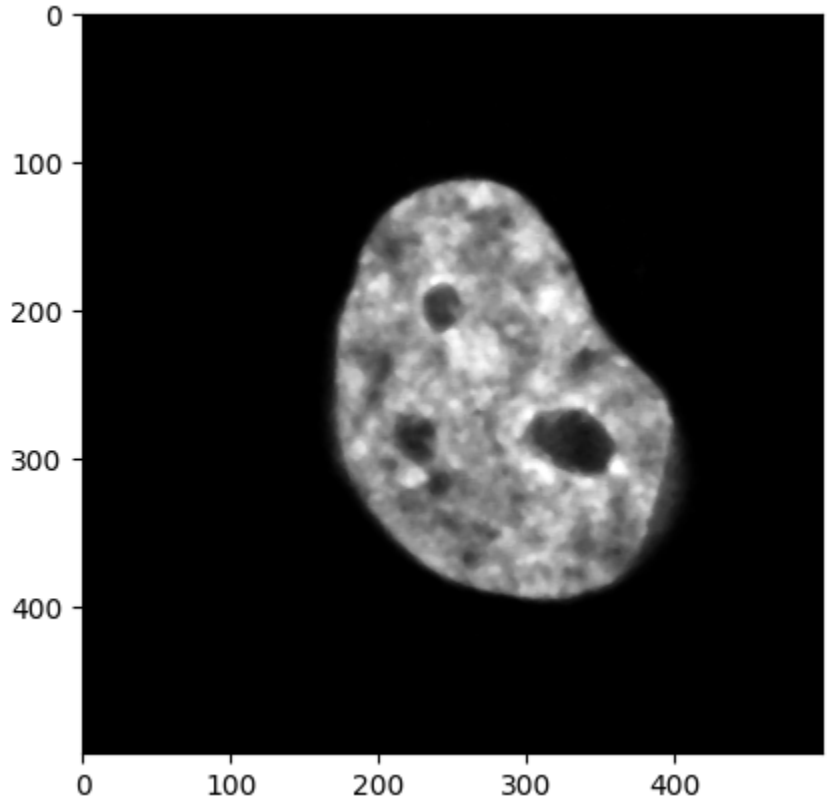
In [512…

```
filtered_median = skimage.filters.median(image_noisy)

plt.subplots(figsize=(10,10))
plt.imshow(filtered_median, cmap = 'gray');
```



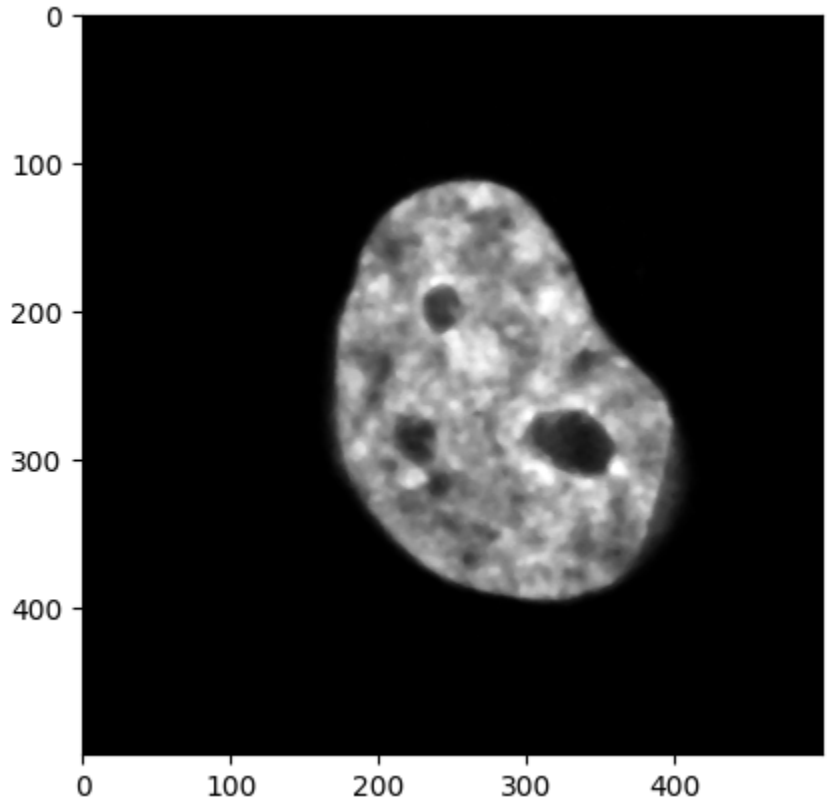
In [513…

```
footprint = np.ones((5,5))
filtered_median = skimage.filters.median(image_noisy, footprint=footprint)
plt.imshow(filtered_median, cmap = 'gray');
```



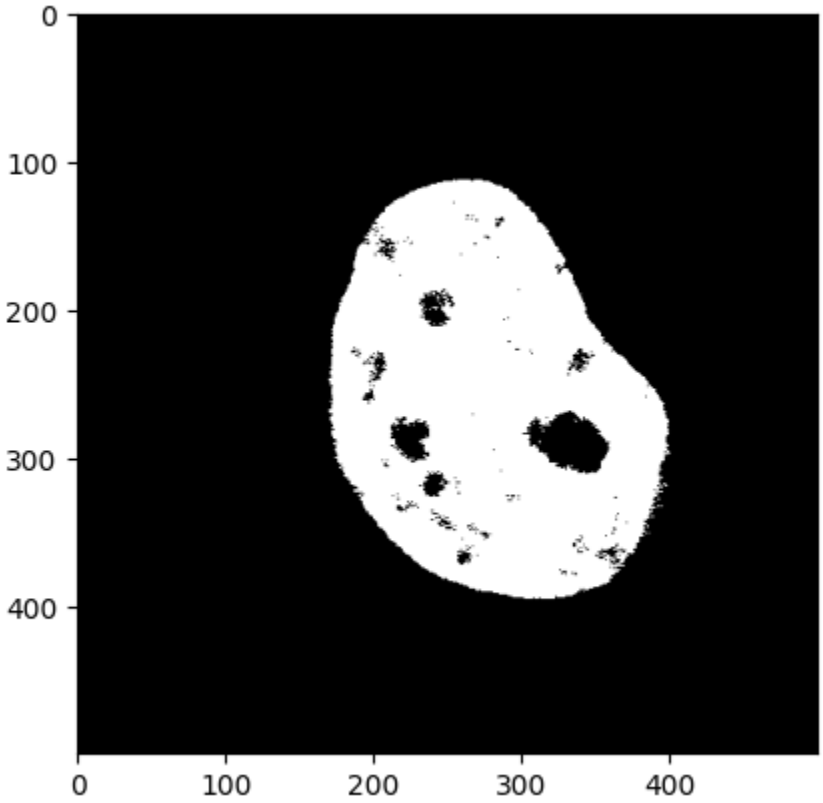
In [514…

```
footprint = skimage.morphology.disk(3)
filtered_median = skimage.filters.median(image_noisy, footprint=footprint)
plt.imshow(filtered_median, cmap = 'gray');
```



In [515…

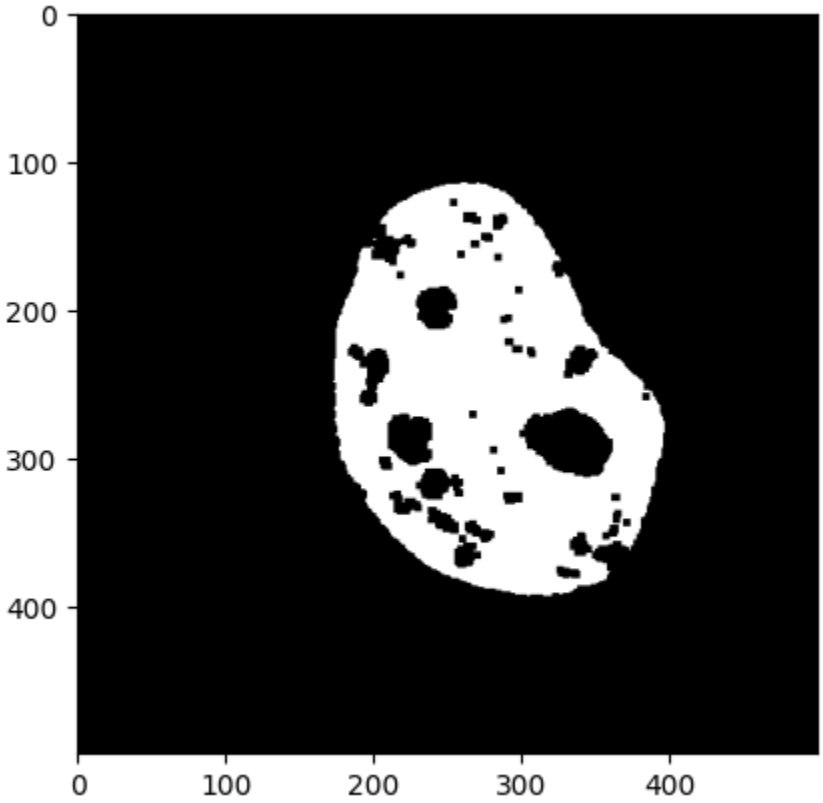
```
mask = image_nuclei > skimage.filters.threshold_otsu(image_nuclei)

plt.imshow(mask, cmap = 'gray');
```



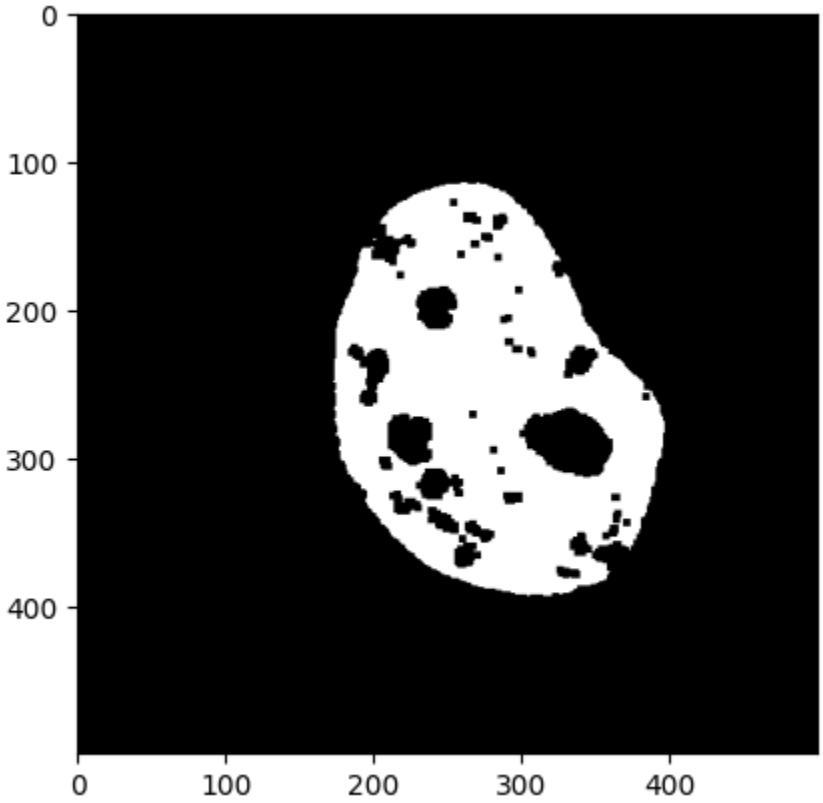
In [516…
```
influence_region = np.ones((5,5))
```

In [517…
```
eroded = skimage.morphology.binary_erosion(mask, footprint=influence_region)
```

In [518…
```
plt.imshow(eroded, cmap = 'gray');
```

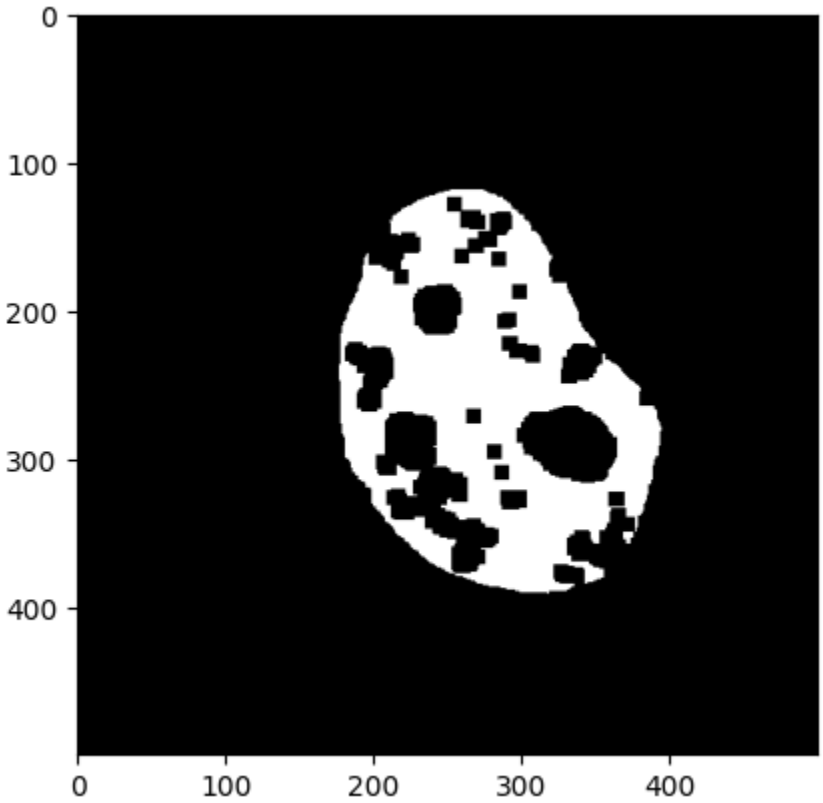


In [519…
```
plt.imshow(eroded, cmap = 'gray');
```



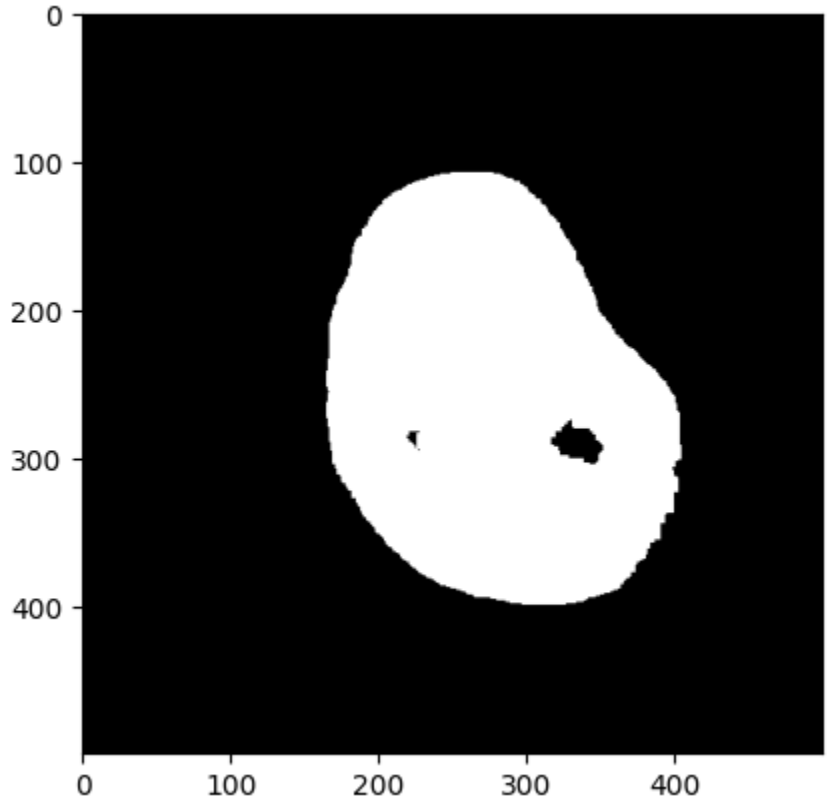
In [520…

```
influence_region = np.ones((10,10))

eroded = skimage.morphology.binary_erosion(mask, footprint=influence_region)

plt.imshow(eroded, cmap = 'gray');
```



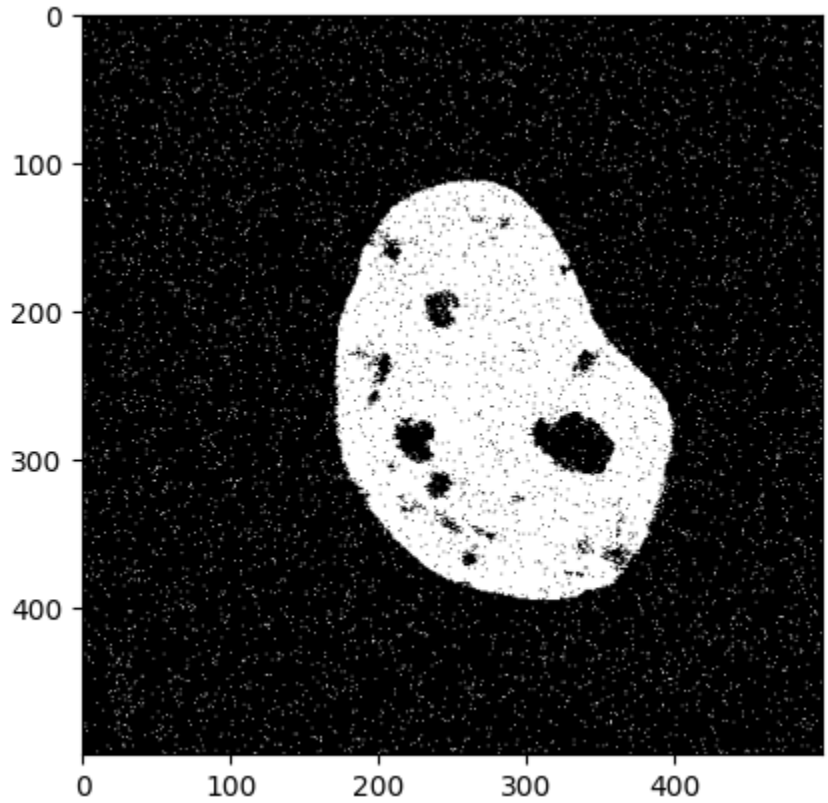
In [521…

```
influence_region = np.ones((10,10))

dilated = skimage.morphology.binary_dilation(mask, footprint=influence_region)

plt.imshow(dilated, cmap = 'gray');
```



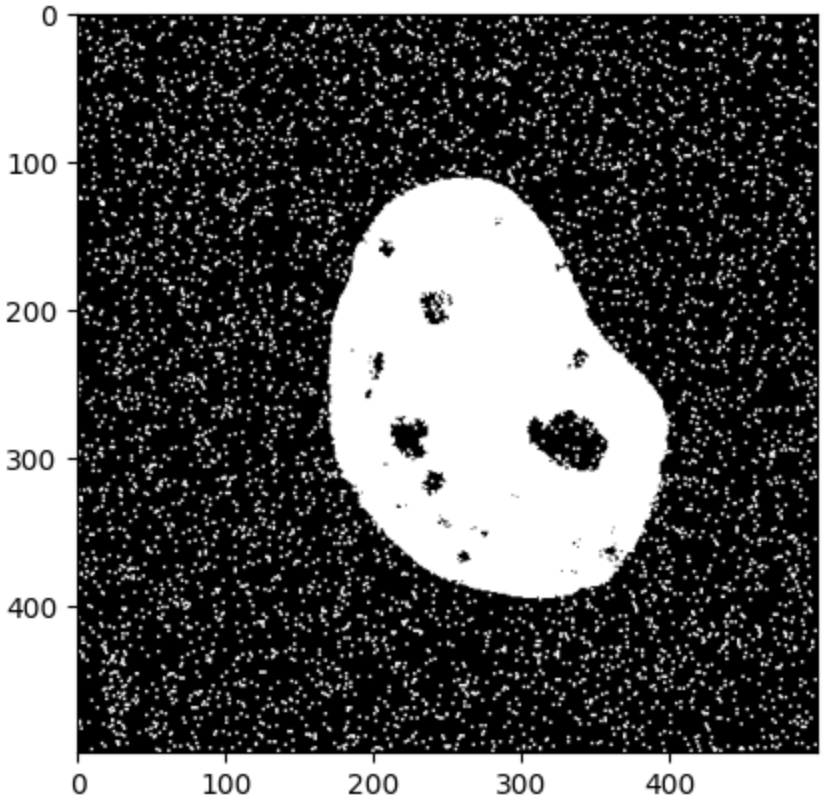
In [522…

```
mask = image_noisy > skimage.filters.threshold_otsu(image_noisy)

plt.imshow(mask, cmap = 'gray');
```



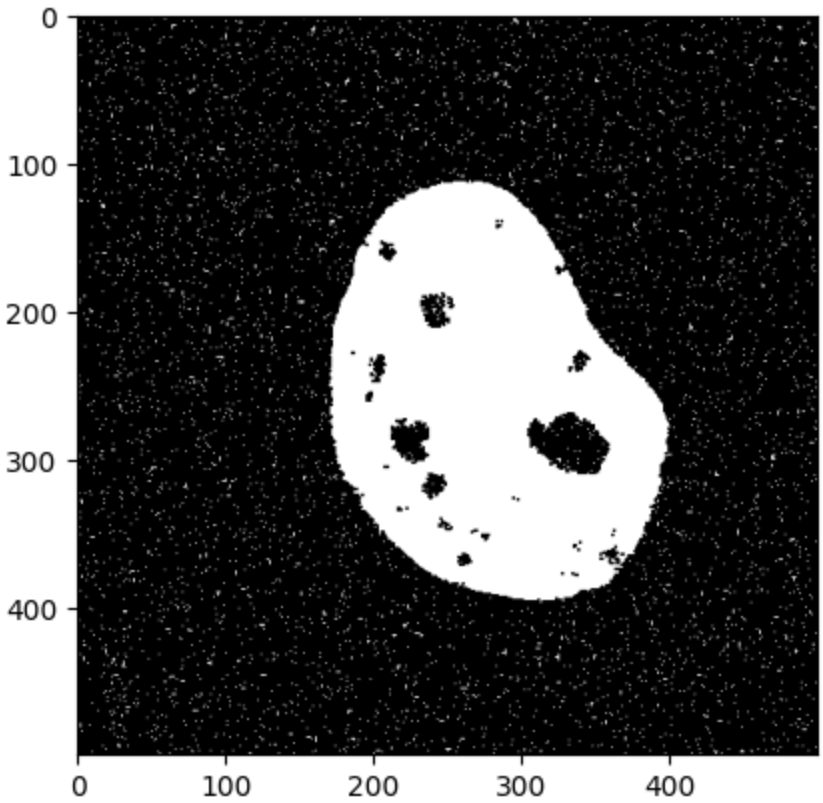
In [523…

```
influence_region = np.ones((2,2))

step1 = skimage.morphology.binary_dilation(mask, footprint=influence_region)

plt.imshow(step1, cmap = 'gray');
```



In [524…

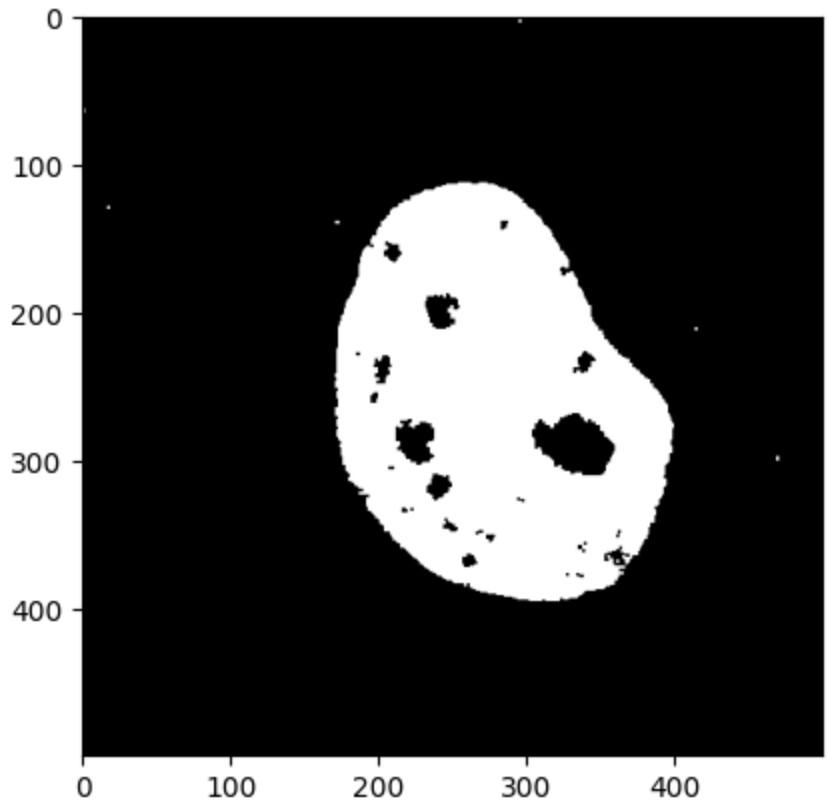
```
influence_region = np.ones((2,2))

step2 = skimage.morphology.binary_erosion(step1, footprint=influence_region)

plt.imshow(step2, cmap = 'gray');
```



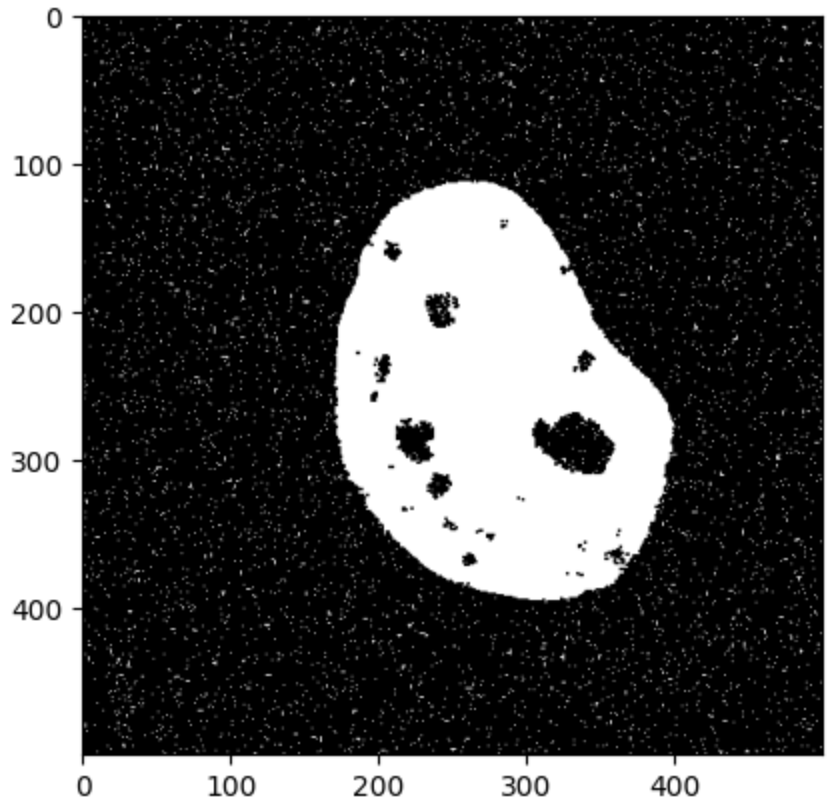
In [525…

```
influence_region = np.ones((2,2))

step3 = skimage.morphology.binary_erosion(step2, footprint=influence_region)
step4 = skimage.morphology.binary_dilation(step3, footprint=influence_region)
```

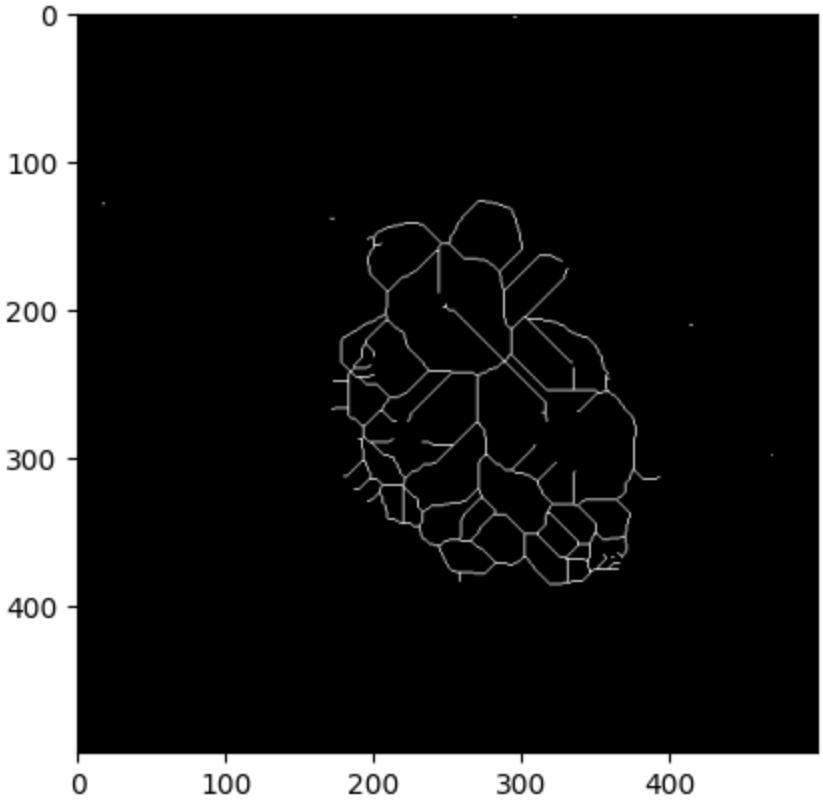
```
plt.imshow(step4, cmap = 'gray');
```



In [526…
```
plt.imshow(skimage.morphology.binary_closing(mask, footprint=influence_region), cmap='gr
```



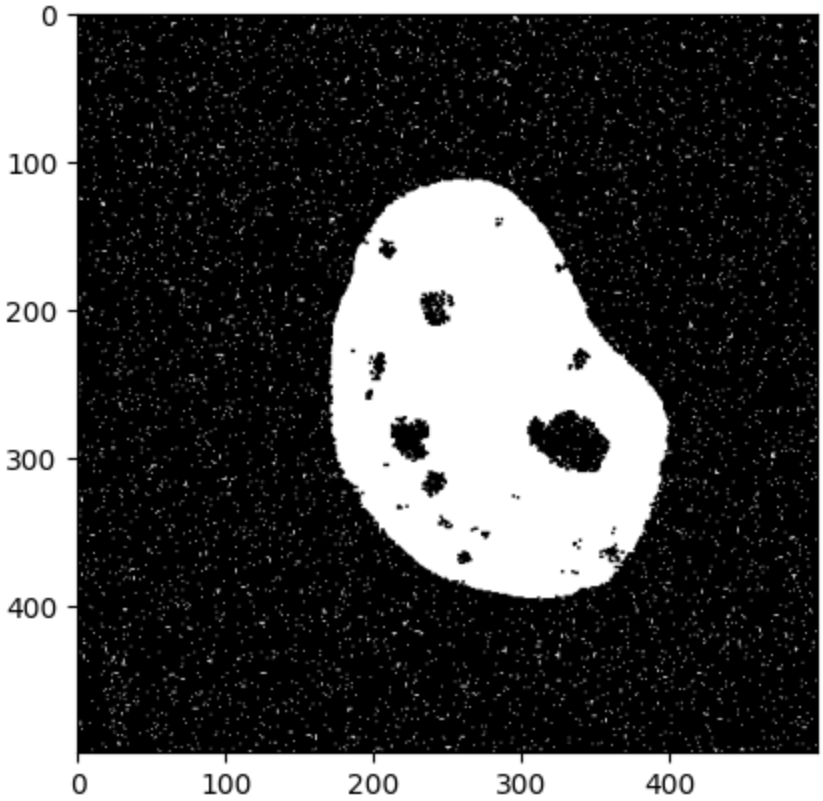
In [527…
```
skeleton = skimage.morphology.skeletonize(step4)

plt.imshow(skeleton, cmap = 'gray');
```



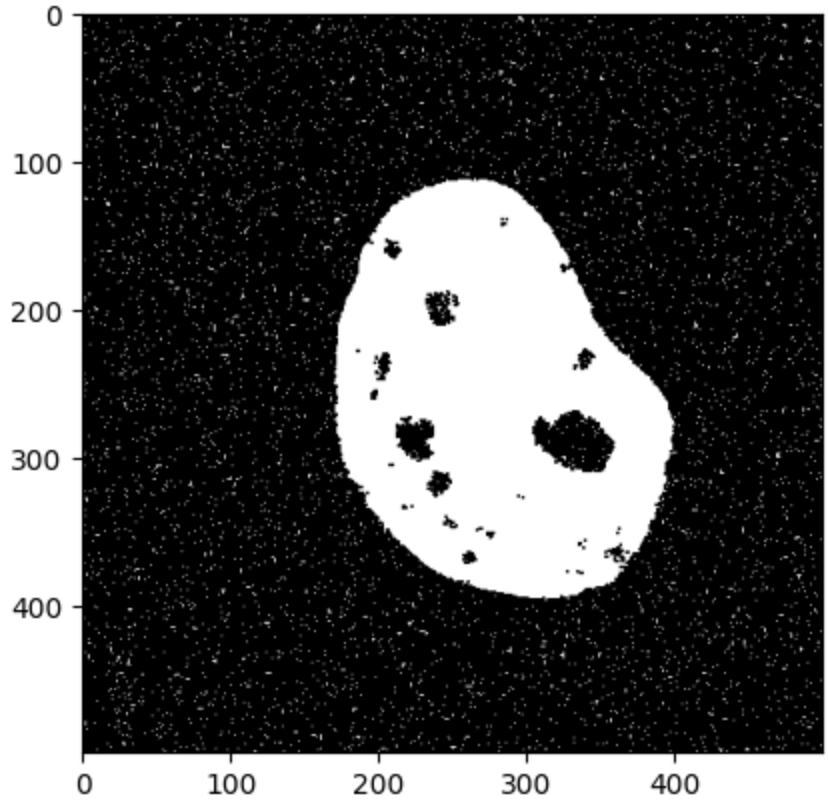
In [528…
```
import scipy.ndimage as ndi
```

In [529…
```
image_closed = skimage.morphology.binary_closing(mask, footprint=influence_region)
```

In [530…
```
plt.imshow(image_closed, cmap='gray');
```

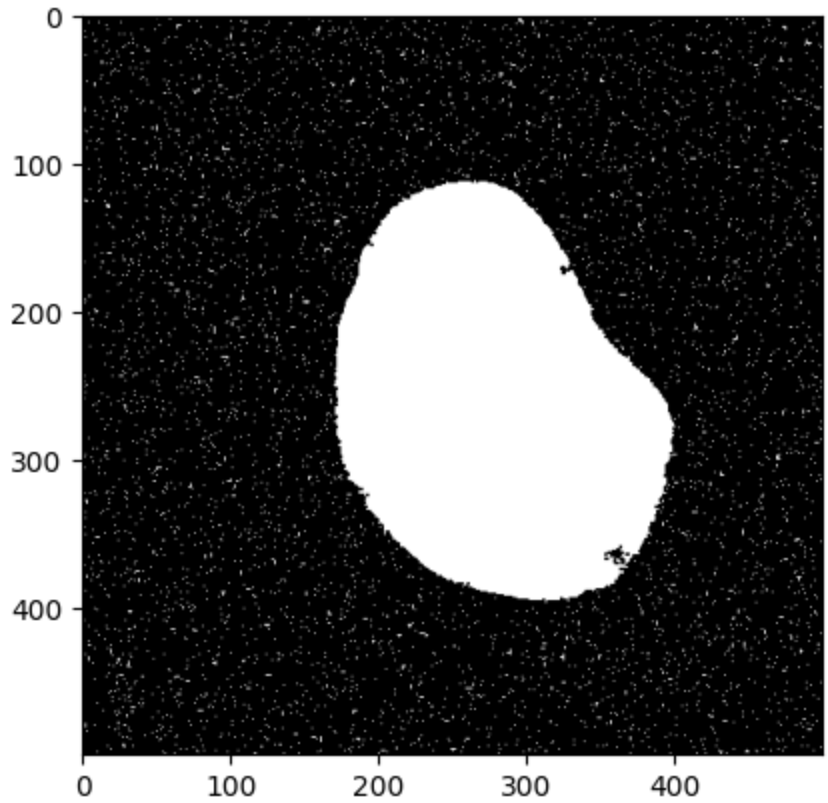


In [531…
```
plt.imshow(image_closed, cmap='gray');
```



In [532…
```
image_fill = ndi.binary_fill_holes(image_closed, skimage.morphology.disk(5))
```

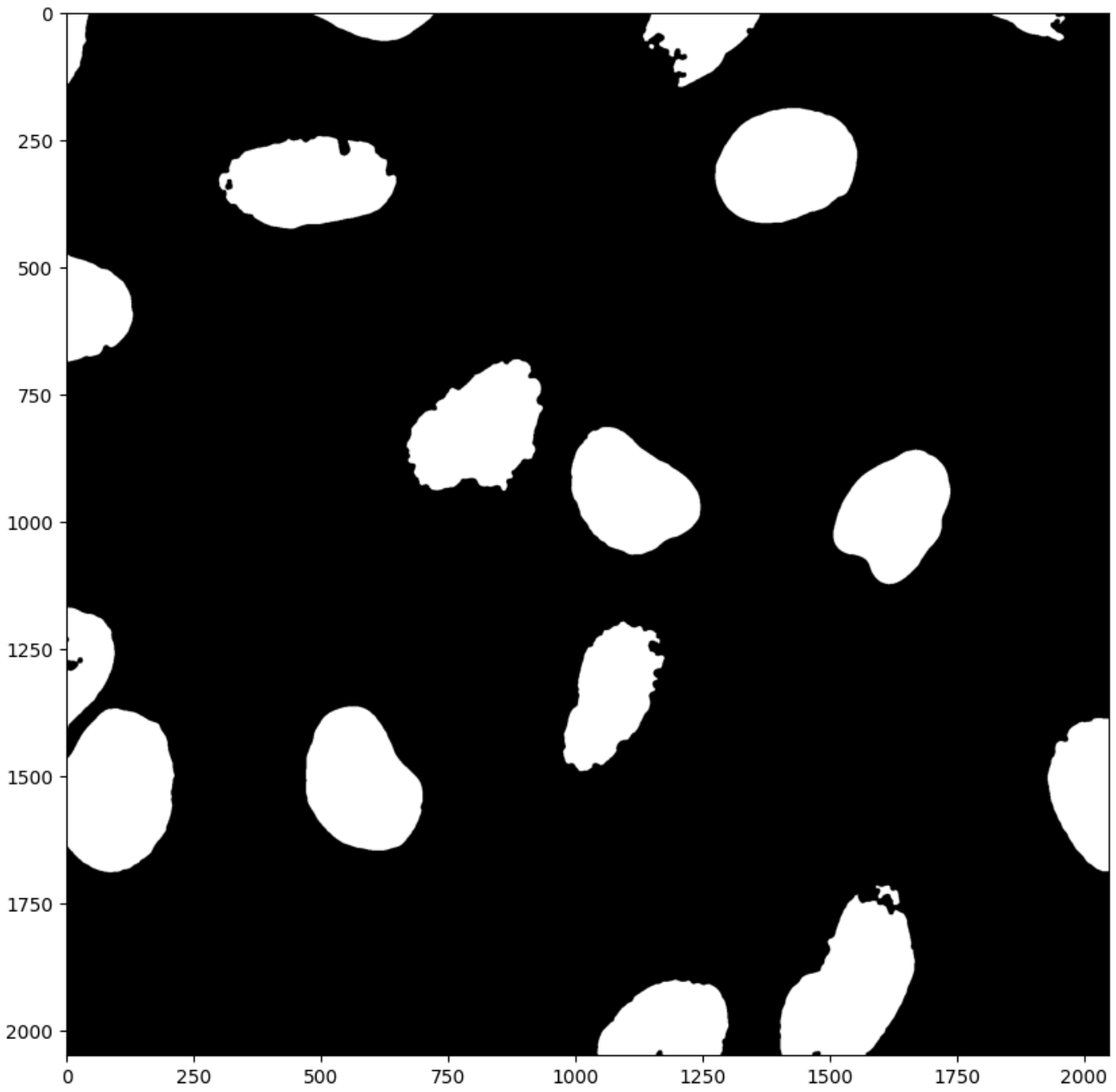
In [533…
```
plt.imshow(image_fill, cmap='gray');
```



In [534…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import skimage
import skimage.io
import skimage.morphology
import scipy.ndimage as ndi
```

In [535…
```
#image_stack = skimage.io.imread('images/46658_784_B12_1.tif')
image_stack = skimage.io.imread('https://github.com/guiwitz/PyImageCourse_beginner/raw/m

image_nuclei = image_stack[:,:,2]#blue channel in RGB
image_signal = image_stack[:,:,1]#green channel in RGB

# filter image
image_nuclei = skimage.filters.median(image_nuclei, skimage.morphology.disk(5))

# create mask and clean-up
mask_nuclei = image_nuclei > skimage.filters.threshold_otsu(image_nuclei)
mask_nuclei = skimage.morphology.binary_closing(mask_nuclei, footprint=skimage.morpholog
mask_nuclei = ndi.binary_fill_holes(mask_nuclei, skimage.morphology.disk(5))
```

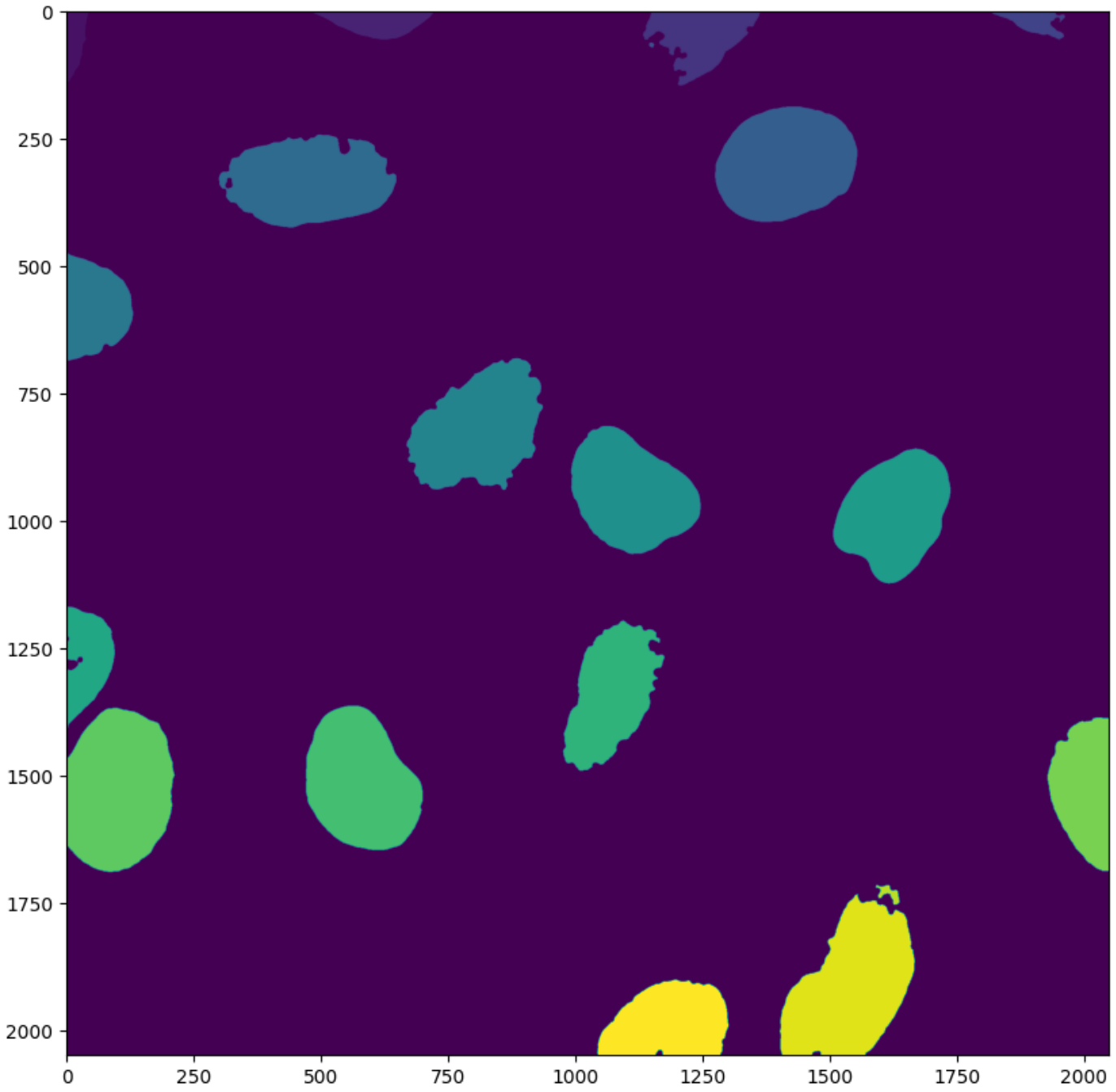
In [536…
```
plt.subplots(figsize=(10,10))
plt.imshow(mask_nuclei, cmap = 'gray');
```



In [537…
```
my_labels = skimage.morphology.label(mask_nuclei)
```

In [538…
```
plt.subplots(figsize=(10,10))
```

```
plt.imshow(my_labels);
```



In [539…
```
my_regions = skimage.measure.regionprops_table(my_labels, properties=('label','area'))
```

In [540…
```
my_regions
```

Out[540]:
```
{'label': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20]),
 'area': array([4.2210e+03, 8.3860e+03, 1.8258e+04, 4.0430e+03, 7.3000e+01,
        4.9056e+04, 4.5671e+04, 2.0537e+04, 4.6853e+04, 4.3277e+04,
        4.0768e+04, 1.5090e+04, 3.5404e+04, 4.7642e+04, 5.4978e+04,
        2.6774e+04, 4.0000e+00, 8.1200e+02, 5.4298e+04, 2.9638e+04])}
```

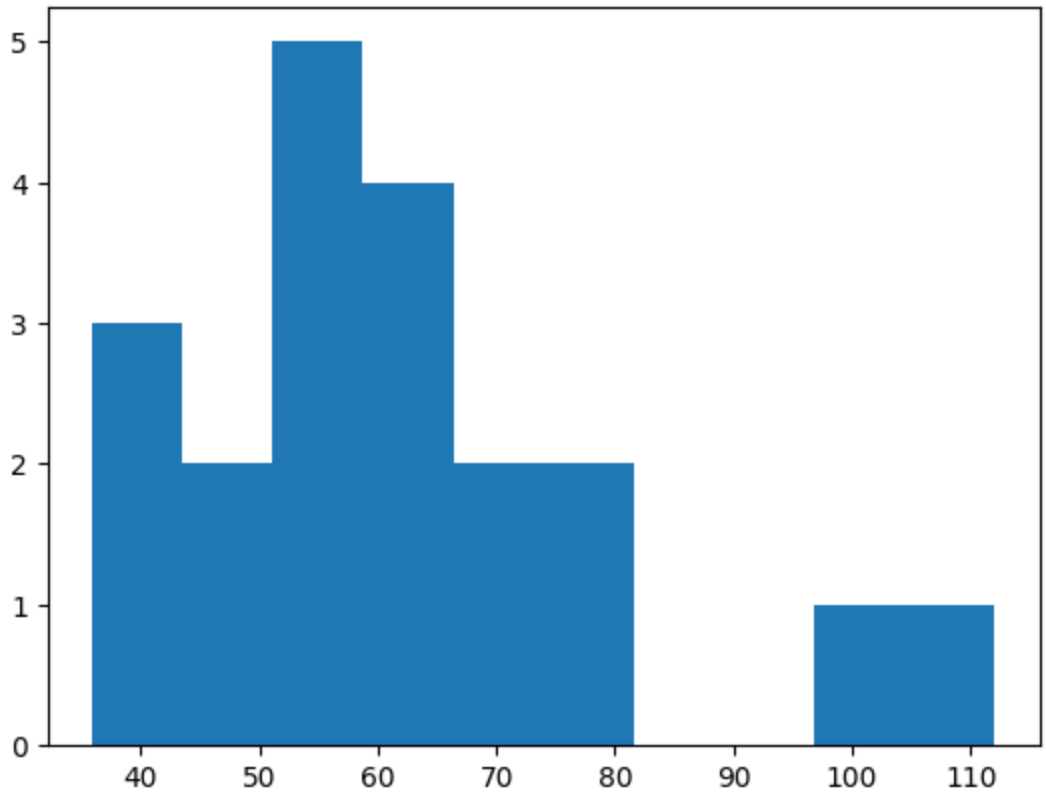
In [541…
```
my_regions = skimage.measure.regionprops_table(
    my_labels,intensity_image=image_signal, properties=('label','area','mean_intensity'))
```

In [542…
```
my_regions
```

Out[542]:
```
{'label': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20]),
 'area': array([4.2210e+03, 8.3860e+03, 1.8258e+04, 4.0430e+03, 7.3000e+01,
        4.9056e+04, 4.5671e+04, 2.0537e+04, 4.6853e+04, 4.3277e+04,
```

```
        4.0768e+04, 1.5090e+04, 3.5404e+04, 4.7642e+04, 5.4978e+04,
        2.6774e+04, 4.0000e+00, 8.1200e+02, 5.4298e+04, 2.9638e+04]),
 'mean_intensity': array([ 79.26510306,  65.99773432,  70.26109103,  63.02671284,
        47.43835616,  53.71450995,  53.8070986 ,  70.45391245,
        66.13772864,  41.4125517 ,  58.67503925,  44.19814447,
        74.77155124,  41.70190168,  35.9041071 ,  53.07787406,
       112.        , 102.9864532 ,  56.11020664,  66.28379108])}
```

In [543…]
```
plt.hist(my_regions['mean_intensity']);
```



In [544…]
```
my_regions['area']
```

Out[544]:
```
array([4.2210e+03, 8.3860e+03, 1.8258e+04, 4.0430e+03, 7.3000e+01,
       4.9056e+04, 4.5671e+04, 2.0537e+04, 4.6853e+04, 4.3277e+04,
       4.0768e+04, 1.5090e+04, 3.5404e+04, 4.7642e+04, 5.4978e+04,
       2.6774e+04, 4.0000e+00, 8.1200e+02, 5.4298e+04, 2.9638e+04])
```

In [545…]
```
selected = my_regions['area'] > 100
selected
```

Out[545]:
```
array([ True,  True,  True,  True, False,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True, False,  True,
        True,  True])
```

In [546…]
```
my_regions['mean_intensity'][selected]
```

Out[546]:
```
array([ 79.26510306,  65.99773432,  70.26109103,  63.02671284,
        53.71450995,  53.8070986 ,  70.45391245,  66.13772864,
        41.4125517 ,  58.67503925,  44.19814447,  74.77155124,
        41.70190168,  35.9041071 ,  53.07787406, 102.9864532 ,
        56.11020664,  66.28379108])
```

In [547…]
```
import pandas as pd
```

In [548…]
```
np.random.seed(42)
my_array = np.random.randint(0,100, (3,5))
my_array
```

Out[548]:
```
array([[51, 92, 14, 71, 60],
```

```
        [20, 82, 86, 74, 74],
        [87, 99, 23,  2, 21]])
```

In [549…
```
pd.DataFrame(my_array)
```

Out[549]:

	0	1	2	3	4
0	51	92	14	71	60
1	20	82	86	74	74
2	87	99	23	2	21

In [550…
```
pd.DataFrame(my_array)
```

Out[550]:

	0	1	2	3	4
0	51	92	14	71	60
1	20	82	86	74	74
2	87	99	23	2	21

In [551…
```
my_df = pd.DataFrame(my_array, columns=['a', 'b', 'c', 'd', 'e'])
my_df
```

Out[551]:

	a	b	c	d	e
0	51	92	14	71	60
1	20	82	86	74	74
2	87	99	23	2	21

In [552…
```
my_df['c']
```

Out[552]:
```
0    14
1    86
2    23
Name: c, dtype: int32
```

In [553…
```
my_df['c'] < 50
```

Out[553]:
```
0     True
1    False
2     True
Name: c, dtype: bool
```

In [554…
```
my_df[my_df['c'] < 50]
```

Out[554]:

	a	b	c	d	e
0	51	92	14	71	60
2	87	99	23	2	21

In [555…
```
my_regions
```

Out[555]:
```
{'label': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20]),
 'area': array([4.2210e+03, 8.3860e+03, 1.8258e+04, 4.0430e+03, 7.3000e+01,
        4.9056e+04, 4.5671e+04, 2.0537e+04, 4.6853e+04, 4.3277e+04,
        4.0768e+04, 1.5090e+04, 3.5404e+04, 4.7642e+04, 5.4978e+04,
        2.6774e+04, 4.0000e+00, 8.1200e+02, 5.4298e+04, 2.9638e+04]),
 'mean_intensity': array([ 79.26510306,  65.99773432,  70.26109103,  63.02671284,
        47.43835616,  53.71450995,  53.8070986 ,  70.45391245,
```

```
 66.13772864,  41.4125517 ,  58.67503925,  44.19814447,
 74.77155124,  41.70190168,  35.9041071 ,  53.07787406,
112.        , 102.9864532 ,  56.11020664,  66.28379108])}
```

In [556…
```
my_regions_df = pd.DataFrame(my_regions)
my_regions_df
```

Out[556]:

	label	area	mean_intensity
0	1	4221.0	79.265103
1	2	8386.0	65.997734
2	3	18258.0	70.261091
3	4	4043.0	63.026713
4	5	73.0	47.438356
5	6	49056.0	53.714510
6	7	45671.0	53.807099
7	8	20537.0	70.453912
8	9	46853.0	66.137729
9	10	43277.0	41.412552
10	11	40768.0	58.675039
11	12	15090.0	44.198144
12	13	35404.0	74.771551
13	14	47642.0	41.701902
14	15	54978.0	35.904107
15	16	26774.0	53.077874
16	17	4.0	112.000000
17	18	812.0	102.986453
18	19	54298.0	56.110207
19	20	29638.0	66.283791

In [557…
```
my_regions_df[my_regions_df['area'] > 100]
```

Out[557]:

	label	area	mean_intensity
0	1	4221.0	79.265103
1	2	8386.0	65.997734
2	3	18258.0	70.261091
3	4	4043.0	63.026713
5	6	49056.0	53.714510
6	7	45671.0	53.807099
7	8	20537.0	70.453912
8	9	46853.0	66.137729
9	10	43277.0	41.412552
10	11	40768.0	58.675039
11	12	15090.0	44.198144
12	13	35404.0	74.771551

13	14	47642.0	41.701902
14	15	54978.0	35.904107
15	16	26774.0	53.077874
17	18	812.0	102.986453
18	19	54298.0	56.110207
19	20	29638.0	66.283791

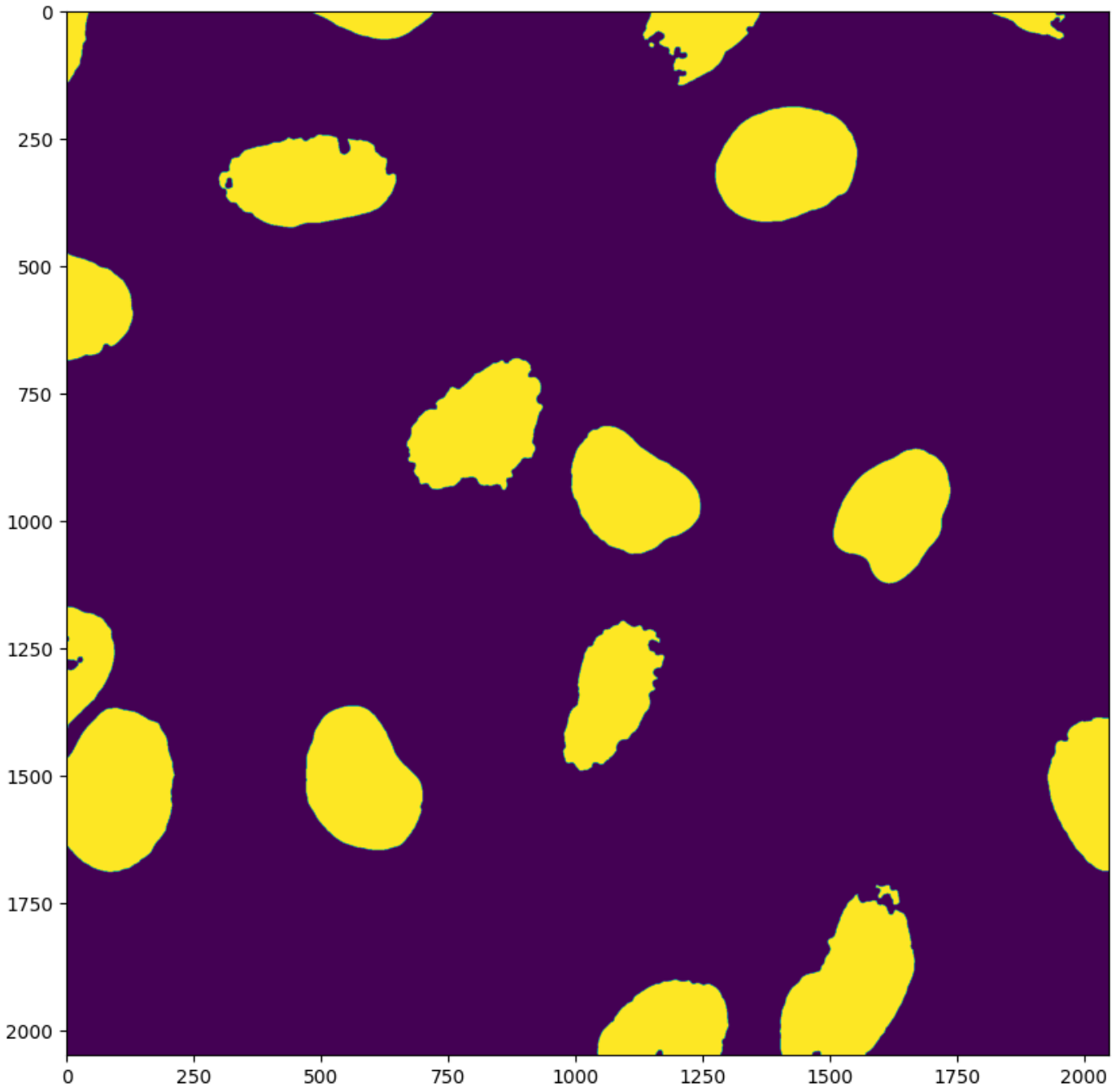
In [558…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import skimage
import skimage.io
import skimage.morphology
from scipy import ndimage as ndi
```

In [559…

```
def my_pipeline(image_path):
    """
    This function extracts information about nuclei found in the third channel of an ima
    loaded at the given path

    Parameters
    ----------
    image_path: str
        path to image

    Returns
    -------
    my_table: pandas dataframe
        dataframe with label, area, mean_intensity and extent information for each nucle

    """

    image_stack = skimage.io.imread(image_path)

    image_nuclei = image_stack[:,:,2]#blue channel in RGB
    image_signal = image_stack[:,:,1]#green channel in RGB

    # filter image
    image_nuclei = skimage.filters.median(image_nuclei, skimage.morphology.disk(5))

    # create mask and clean-up
    mask_nuclei = image_nuclei > skimage.filters.threshold_otsu(image_nuclei)
    mask_nuclei = skimage.morphology.binary_closing(mask_nuclei, footprint=skimage.morph
    mask_nuclei = ndi.binary_fill_holes(mask_nuclei, skimage.morphology.disk(5))

    # label image
    my_labels = skimage.morphology.label(mask_nuclei)

    # measure
    my_regions = skimage.measure.regionprops_table(my_labels, image_signal, properties=(

    plt.subplots(figsize=(10,10))
    plt.imshow(mask_nuclei)
    plt.show()

    my_table = pd.DataFrame(my_regions)

    return my_table
```

In [560…
```
table = my_pipeline('https://github.com/guiwitz/PyImageCourse_beginner/raw/master/images
```



In [561…
```
table
```

Out[561]:

	label	area	mean_intensity
0	1	4221.0	79.265103
1	2	8386.0	65.997734
2	3	18258.0	70.261091
3	4	4043.0	63.026713
4	5	73.0	47.438356
5	6	49056.0	53.714510
6	7	45671.0	53.807099
7	8	20537.0	70.453912
8	9	46853.0	66.137729
9	10	43277.0	41.412552

10	11	40768.0	58.675039
11	12	15090.0	44.198144
12	13	35404.0	74.771551
13	14	47642.0	41.701902
14	15	54978.0	35.904107
15	16	26774.0	53.077874
16	17	4.0	112.000000
17	18	812.0	102.986453
18	19	54298.0	56.110207
19	20	29638.0	66.283791

In [562…

```
def my_pipeline(image_path, do_plotting=False):
    """
    This function extracts information about nuclei found in the third channel of an ima
    loaded at the given path

    Parameters
    ----------
    image_path: str
        path to image
    do_plotting: bool
        show segmentation or not

    Returns
    -------
    my_table: pandas dataframe
        dataframe with label, area, mean_intensity and extent information for each nucle

    """

    image_stack = skimage.io.imread(image_path)

    image_nuclei = image_stack[:,:,2]#blue channel in RGB
    image_signal = image_stack[:,:,1]#green channel in RGB

    # filter image
    image_nuclei = skimage.filters.median(image_nuclei, skimage.morphology.disk(5))

    # create mask and clean-up
    mask_nuclei = image_nuclei > skimage.filters.threshold_otsu(image_nuclei)
    mask_nuclei = skimage.morphology.binary_closing(mask_nuclei, selem=skimage.morpholog
    mask_nuclei = ndi.binary_fill_holes(mask_nuclei, skimage.morphology.disk(5))

    # label image
    my_labels = skimage.morphology.label(mask_nuclei)

    # measure
    my_regions = skimage.measure.regionprops_table(my_labels, image_signal, properties=(

    if do_plotting:
        plt.subplots(figsize=(10,10))
        plt.imshow(mask_nuclei)
        plt.show()

    my_table = pd.DataFrame(my_regions)

    return my_table
```

In [565…
```python
import requests
from PIL import Image
from io import BytesIO

def my_pipeline(image_url):
    try:
        # Send a GET request to the URL
        response = requests.get(image_url)

        # Check if the request was successful
        if response.status_code == 200:
            # Read the image from the response content
            image = Image.open(BytesIO(response.content))

            # Process the image as needed
            # For example, resizing or applying filters
            # Replace the following lines with your actual image processing operations
            resized_image = image.resize((new_width, new_height))
            # processed_image = apply_filters(resized_image)

            # Store the processed image in the 'table' variable
            table = resized_image

            return table
        else:
            print("Failed to retrieve the image. Status code:", response.status_code)
            return None
    except Exception as e:
        print("Error occurred:", e)
        return None

# Call the pipeline function with the image URL
image_url = 'https://github.com/guiwitz/PyImageCourse_beginner/raw/master/images/46658_7
table = my_pipeline(image_url)

# Check if the 'table' variable contains the processed image
if table is not None:
    print("Image processing successful.")
else:
    print("Image processing failed.")
```

```
Error occurred: name 'new_width' is not defined
Image processing failed.
```

In [566…
```python
files_to_analyze = [
    'https://github.com/guiwitz/PyImageCourse_beginner/raw/master/images/46658_784_B12_1
    'https://github.com/guiwitz/PyImageCourse_beginner/raw/master/images/27897_273_C8_2.
]
```

In [571…
```python
from cellpose import models
```

In [572…
```python
import skimage.io
import matplotlib.pyplot as plt
```

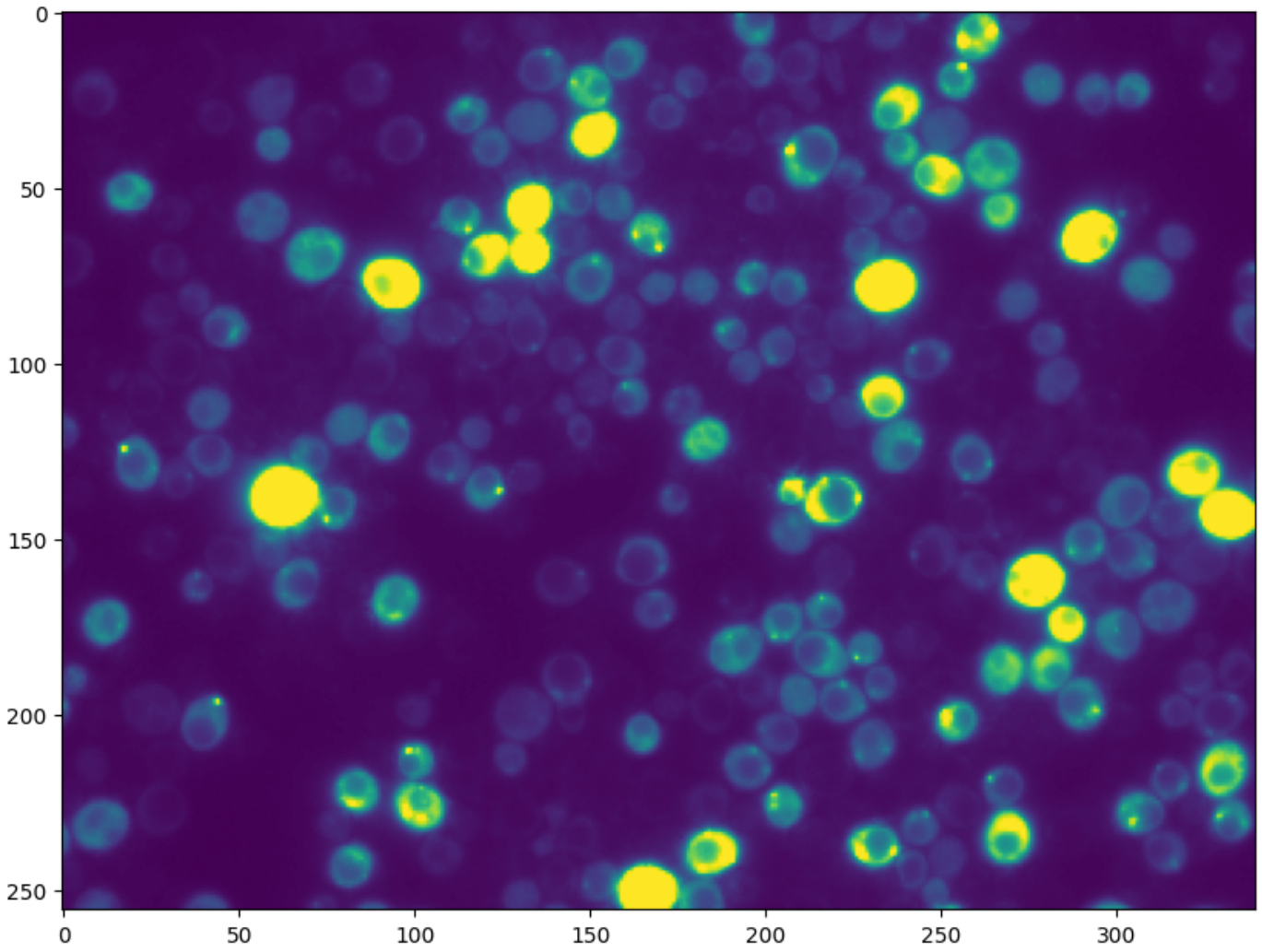
In [574…
```python
image = skimage.io.imread('https://cildata.crbs.ucsd.edu/media/images/13901/13901.tif')
```

In [575…
```python
image.shape
```

Out[575]:
```
(1024, 1360)
```

In [577…
```python
image = image[::4, :: 4]
```

In [578…
```
fig, ax = plt.subplots(figsize=(10,10))
ax.imshow(image);
```

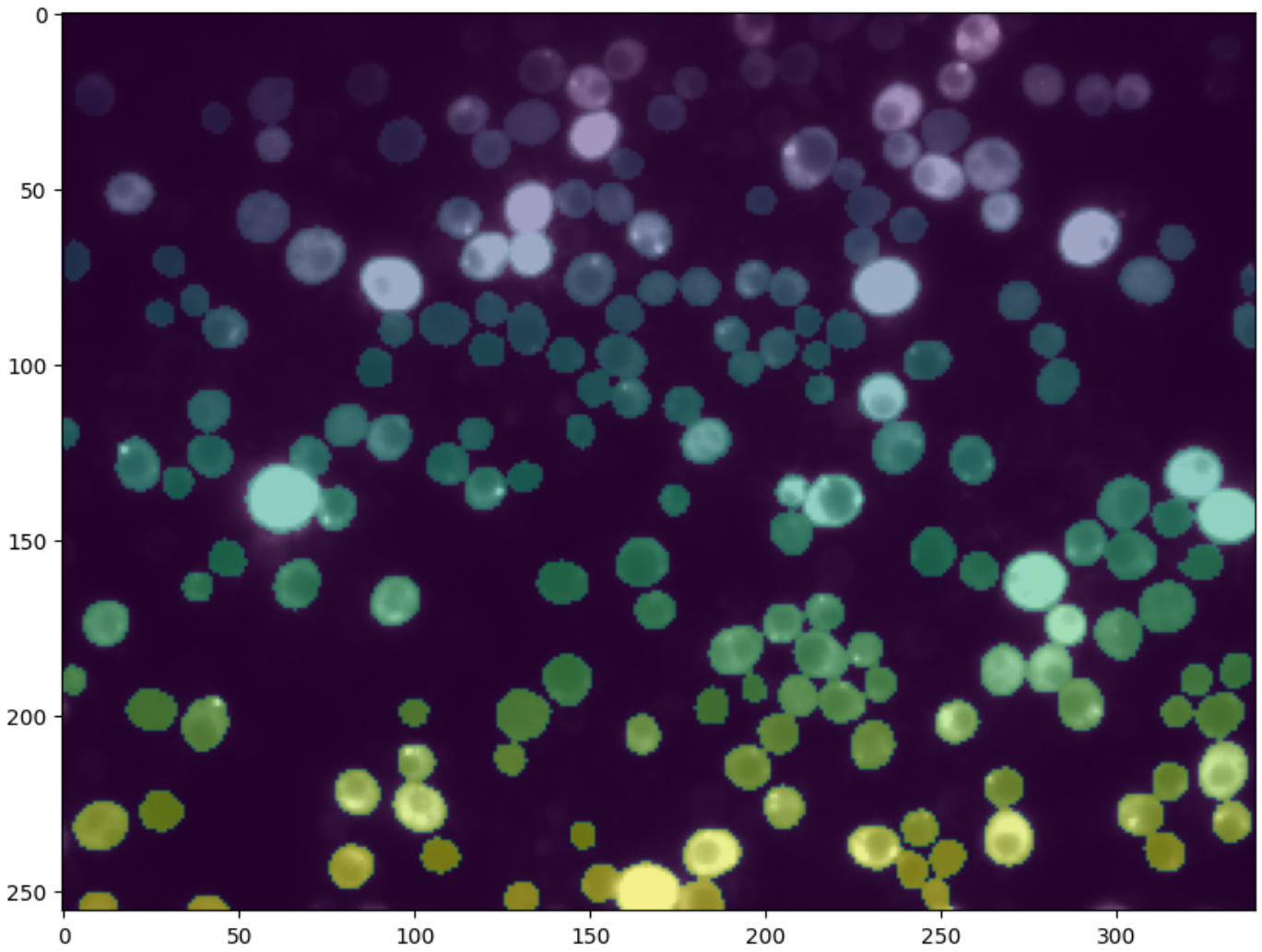


In [579…
```
model = models.Cellpose(model_type='cyto')
```

```
100%|██████████| 25.3M/25.3M [00:25<00:00, 1.04MB/s]
100%|██████████| 5.23k/5.23k [00:00<00:00, 5.34MB/s]
```

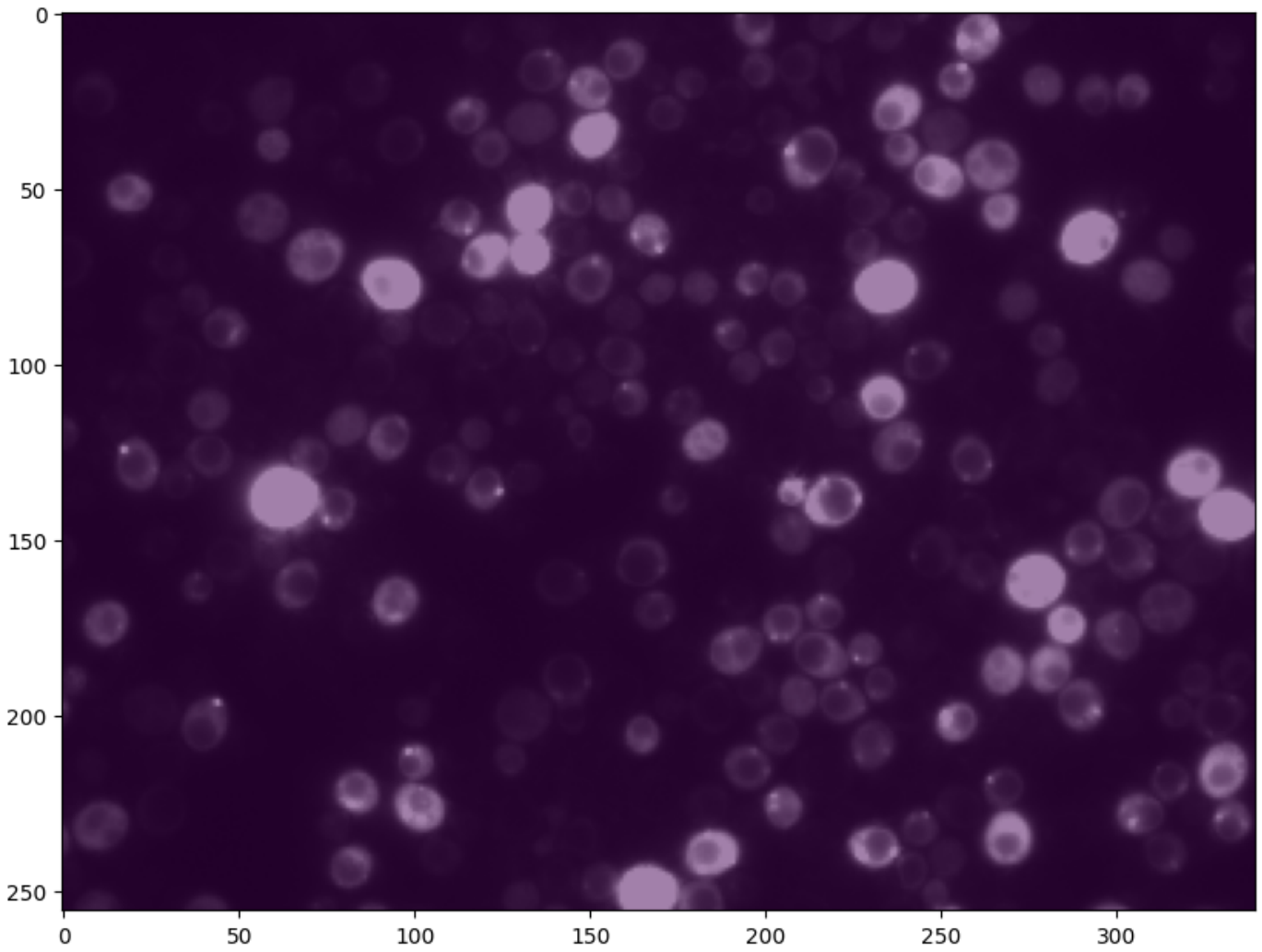
In [580…
```
channels=[[1,3]]
masks, flows, styles, diams = model.eval(image, diameter=None, channels=channels)
```

In [582…
```
fig, ax = plt.subplots(figsize=(10,10))
ax.imshow(image[:,:], cmap='gray')
ax.imshow(masks, alpha=0.5);
```



In [583…
```
masks, flows, styles, diams = model.eval(image, diameter=100, channels=channels)
```

In [585…
```
fig, ax = plt.subplots(figsize=(10,10))
ax.imshow(image[:,:], cmap='gray')
ax.imshow(masks, alpha=0.5);
```



In [586…
```
model = models.Cellpose(model_type='nuclei')

channels = [3,0]
masks, flows, styles, diams = model.eval(image, diameter=100, channels=channels)
```

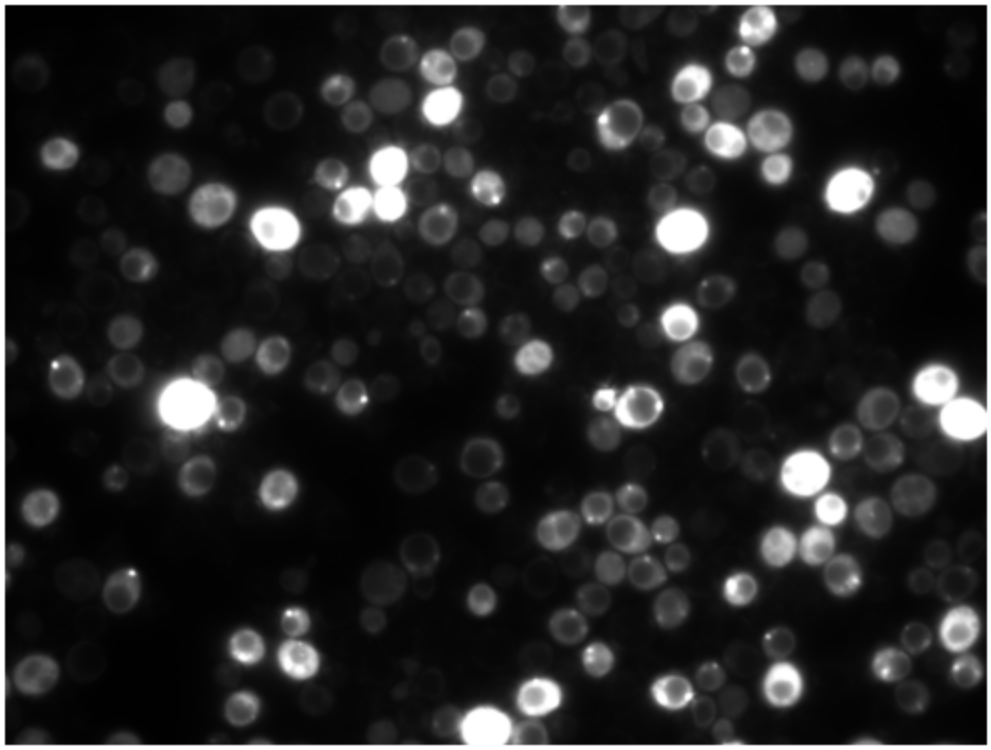
```
100%|██████████| 25.3M/25.3M [00:24<00:00, 1.06MB/s]
100%|██████████| 3.54k/3.54k [00:00<00:00, 1.78MB/s]
```

In [587…
```
import napari
from napari.utils import nbscreenshot
```

In [588…
```
viewer = napari.Viewer()
```

In [605…
```
import requests
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from io import BytesIO

# Fetch the image data from the URL
url = 'https://cildata.crbs.ucsd.edu/media/images/13901/13901.tif'
response = requests.get(url)
if response.status_code == 200:
    # Read the image data using PIL
    image = Image.open(BytesIO(response.content))

    # Convert the image to numpy array
    image_array = np.array(image)
    # print(image_array)
```

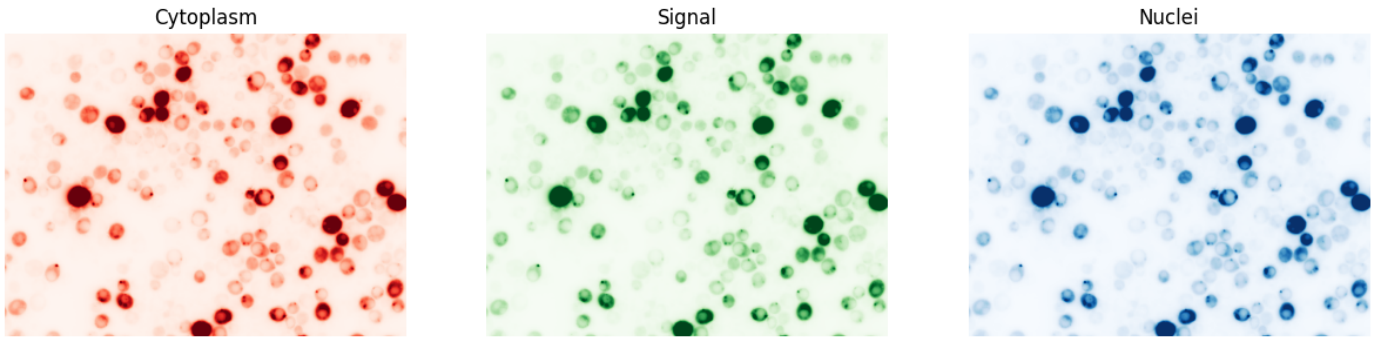
```
    # Display the image using Matplotlib
    plt.imshow(image_array, cmap='gray')
    plt.axis('off')  # Hide axis
    plt.show()
else:
    print("Failed to fetch the image. Status code:", response.status_code)
```



In [608…

```
import requests
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from io import BytesIO

# Fetch the image data from the URL
url = 'https://cildata.crbs.ucsd.edu/media/images/13901/13901.tif'
response = requests.get(url)
if response.status_code == 200:
    # Read the image data using PIL
    image = Image.open(BytesIO(response.content))

    # Convert the image to numpy array
    image_array = np.array(image)

    # Split the image into channels
    cytoplasm = image_array[:,:]
    signal = image_array[:,:]
    nuclei = image_array[:,:]

    # Display each channel with the desired colormap using Matplotlib
    fig, axes = plt.subplots(1, 3, figsize=(15, 5))
    axes[0].imshow(cytoplasm, cmap='Reds')
    axes[0].set_title('Cytoplasm')
    axes[0].axis('off')
    axes[1].imshow(signal, cmap='Greens')
    axes[1].set_title('Signal')
    axes[1].axis('off')
    axes[2].imshow(nuclei, cmap='Blues')
    axes[2].set_title('Nuclei')
    axes[2].axis('off')
    plt.show()
```

```
else:
    print("Failed to fetch the image. Status code:", response.status_code)
```



In [610…
```
viewer.add_labels(masks);
```

In [614…
```
import skimage
from magicgui import magicgui
from napari.types import ImageData, LabelsData
```

In [615…
```
def my_thresholder(image: ImageData, th_level: int=0) -> LabelsData:

    label_image = image > th_level
    label_image = skimage.morphology.label(label_image)

    return(label_image)

thresholding_widget = magicgui(my_thresholder)
```

In [1]:
```
import tkinter as tk

class ThresholdingWidget(tk.Toplevel):
    def __init__(self, parent):
        super().__init__(parent)
        self.title('Thresholding Widget')
        self.geometry('300x200')

        self.label = tk.Label(self, text='Thresholding Parameters')
        self.label.pack(pady=10)

        self.threshold_button = tk.Button(self, text='Apply Threshold')
        self.threshold_button.pack(pady=10)

class MyApplication:
    def __init__(self, root):
        self.root = root
        self.root.title('My Application')
        self.root.geometry('800x600')

        # Create the thresholding widget
        self.thresholding_widget = ThresholdingWidget(root)

def main():
    root = tk.Tk()
    app = MyApplication(root)
    root.mainloop()

if __name__ == '__main__':
    main()
```