Dot Analysis 1.0

Initialization

Some of the cells in this notebook are hidden by default including the next 4. They include details not necessary to user inspection of a given image analysis job.
You can open these cells by double-clicking on the arrowed vertical line to the right of the respective title.

To quit the session and clear all variables, select Evaluation->Quit Kernel->Local. To delete all output, select Cell->Delete all output. These two commands may help in case Mathematica gets stuck on an improper evaluation if this notebook is modified. The full analysis job may take a few gigabytes of memory--retry on a more powerful computer if the workflow below does not complete properly.

To run an entire image analysis job, click the noninteractive analysis button below. If you have not set any global variables yet, you will be asked for a JSON file to load. See provided JSON examples. Individual cells below may be rerun or manipulated to see how results change with different parameters.

For users experienced in Mathematica, to run interactive analysis, you can run each cell sequentially by pressing shift-enter. To see all the code in this notebook, select all the cells and select Cell->Cell Properties->Open. Remember to run all cells below the modifications being made to register all updates in your results; or just rerun the whole notebook, since parameters are stored globally.

In[37]:= **Click here to run complete analysis** ;

Image processing

Colocalization

Graphics

Variables

Load parameters and images

Parameters

Here we display the current of parameters (it dynamically updates as parameters are modified). JSON files may be used to load or save the parameters. The JSON file which is loaded is not modified.

In[73]:=
```
If[! ListQ[PATHS], askforfile[]]; (* if parameters have not been set *)
```

In[74]:=
```
Print[CURRENTPARAMS]
Row[{open, save}]
```

File paths

BRAF-1.tif
BRAF-2.tif

Global parameters

Pixel dimensions	0.0624	0.0624	0.42
Image scaling	1	1	1
Flip image horizontally	True		
Blur standard deviation	0.2	0.2	
Blur extent	1	1	
Background subtraction length	2.5	2.5	
Background subtraction coefficient	1	1	
Min pixel threshold	0.003	0.012	
Max pixel threshold	1	1	
Watershed threshold	0.15	0.15	

Channel parameters

	Channel 1	Channel 2	
Dot intensity definition	1		
Dot intensity threshold	0.01	0.005	
Coordinate offset	0	0	0
Keypoint alignment	False		
Colocalization distance thresholds	0.22	0.42	0.22
Colocalization mode	2		
Dot counts	129	136	110
Colocalization fractions	0	0	0
Notes			

Out[75]=

Open JSON	Save JSON

Import data

If not loading from JSON, the variables in PARAMS should be set before proceeding. Load the images now.

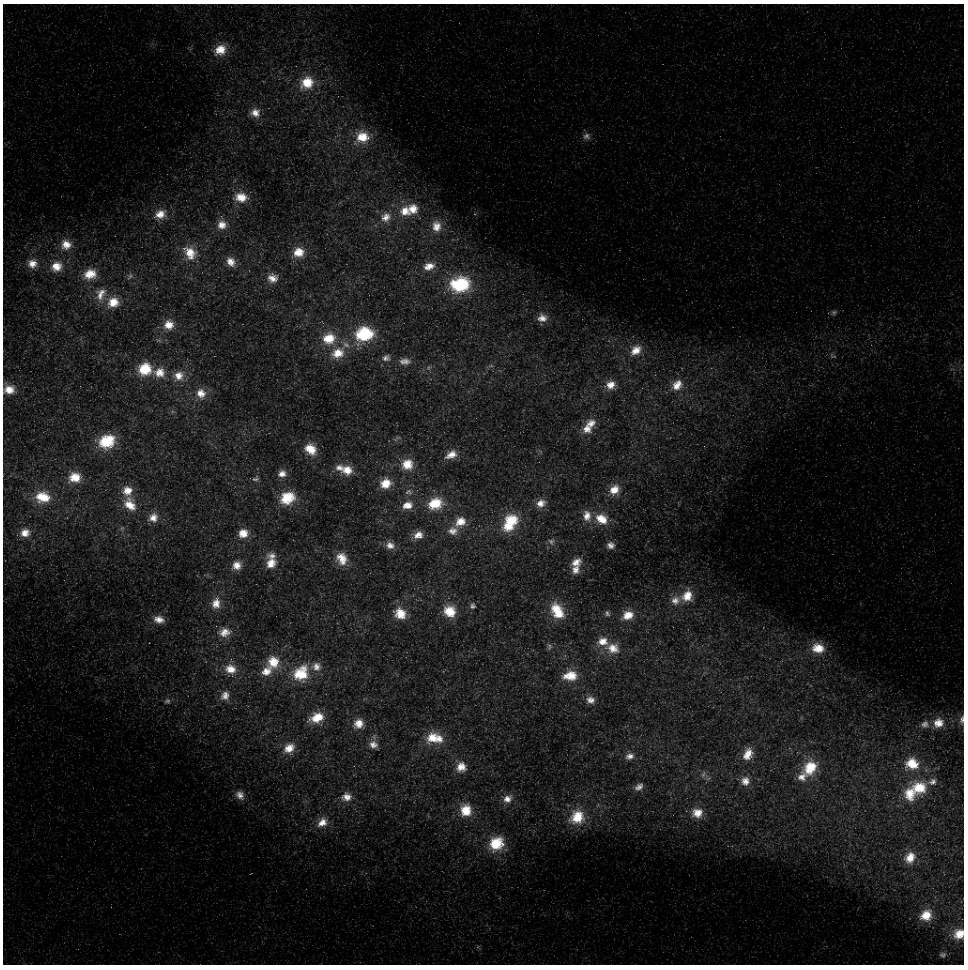
In[76]:=
```
If[! CheckPaths[PATHS], askfortiffs[]];
(* Variables not set or the PATHS don't exist *)
```

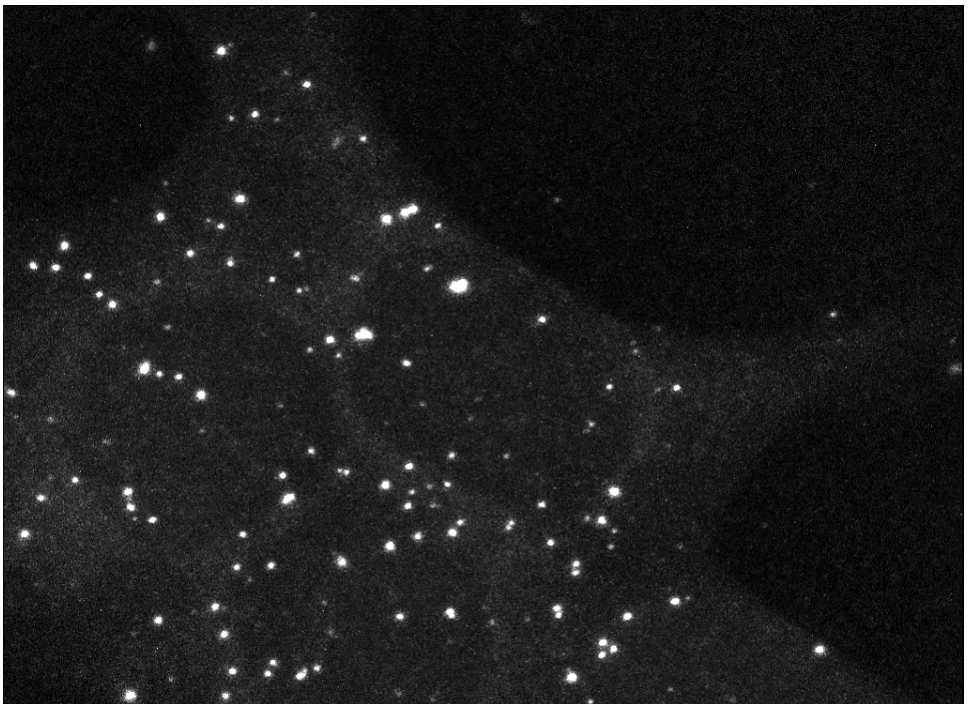
In[77]:=
```
If[! CheckPaths[PATHS], stop[]];
```

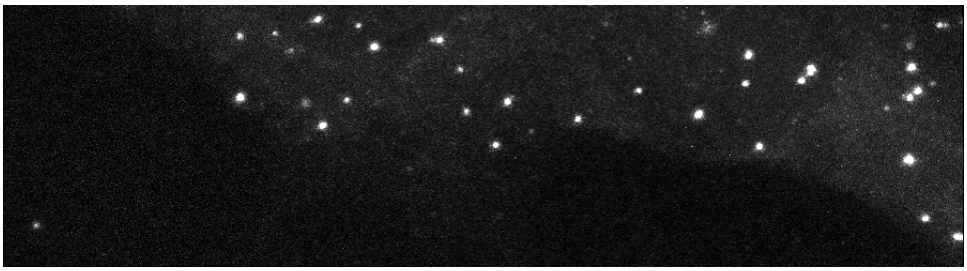
In[78]:=
```
IMG0 = LoadImage[Import[FindFile[#]], FLIP, {1, 1, 1}] & /@ PATHS;
(* Load image and flip horizontal dimension if desired *)
Column[{Text[Style["Unmodified images from disk (channels 1 and 2)", 20, Bold]],
  Row[Show[Proj[#, 5], ImageSize → 500] & /@ IMG0, Spacer[1]]}]
```

Unmodified images from disk (channels 1 and 2)

Out[79]=





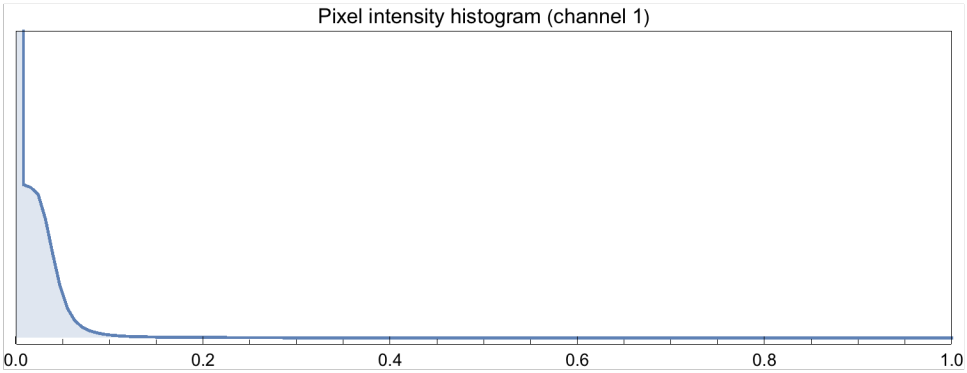


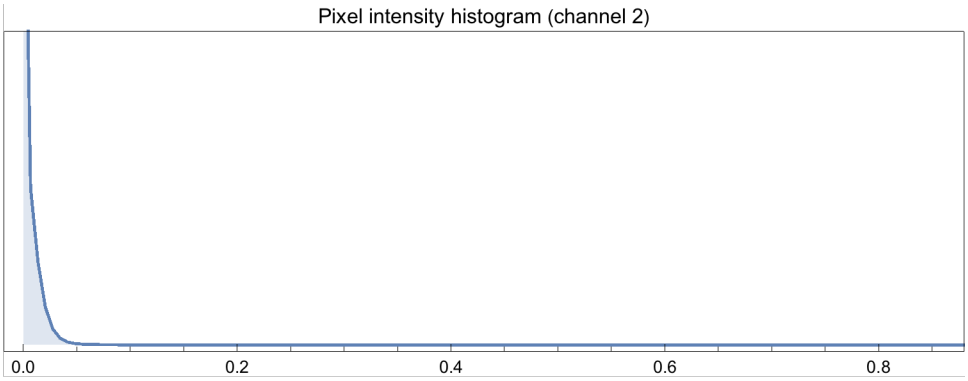
Run analysis

Implementation

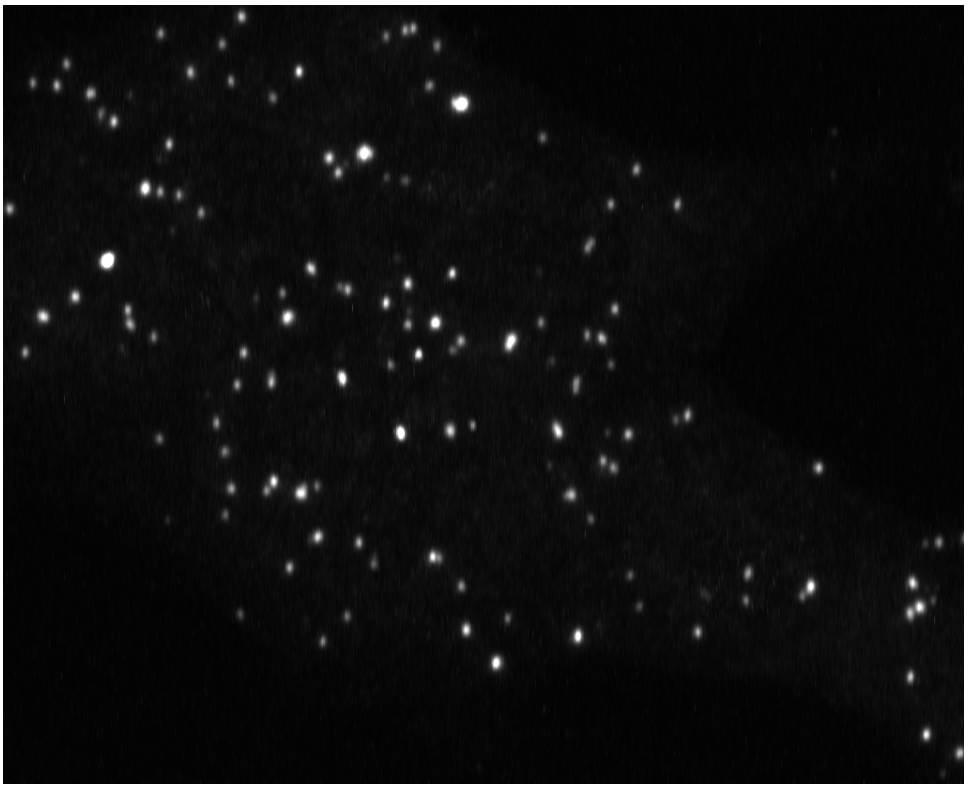
Step 1: Blur noise

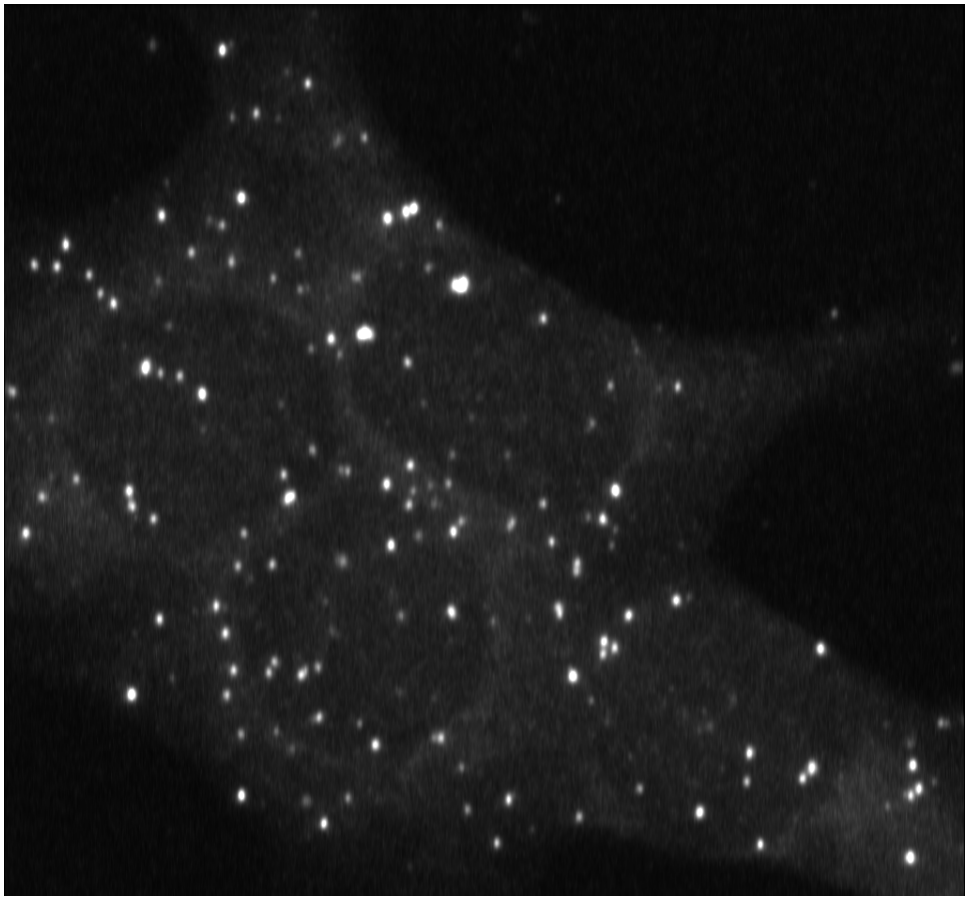
In[82]:=
```
UnblurredIMG = AdjustImages[IMG0];
IMG = BlurNoise[UnblurredIMG];
```





Blurred images for channels 1 and 2:





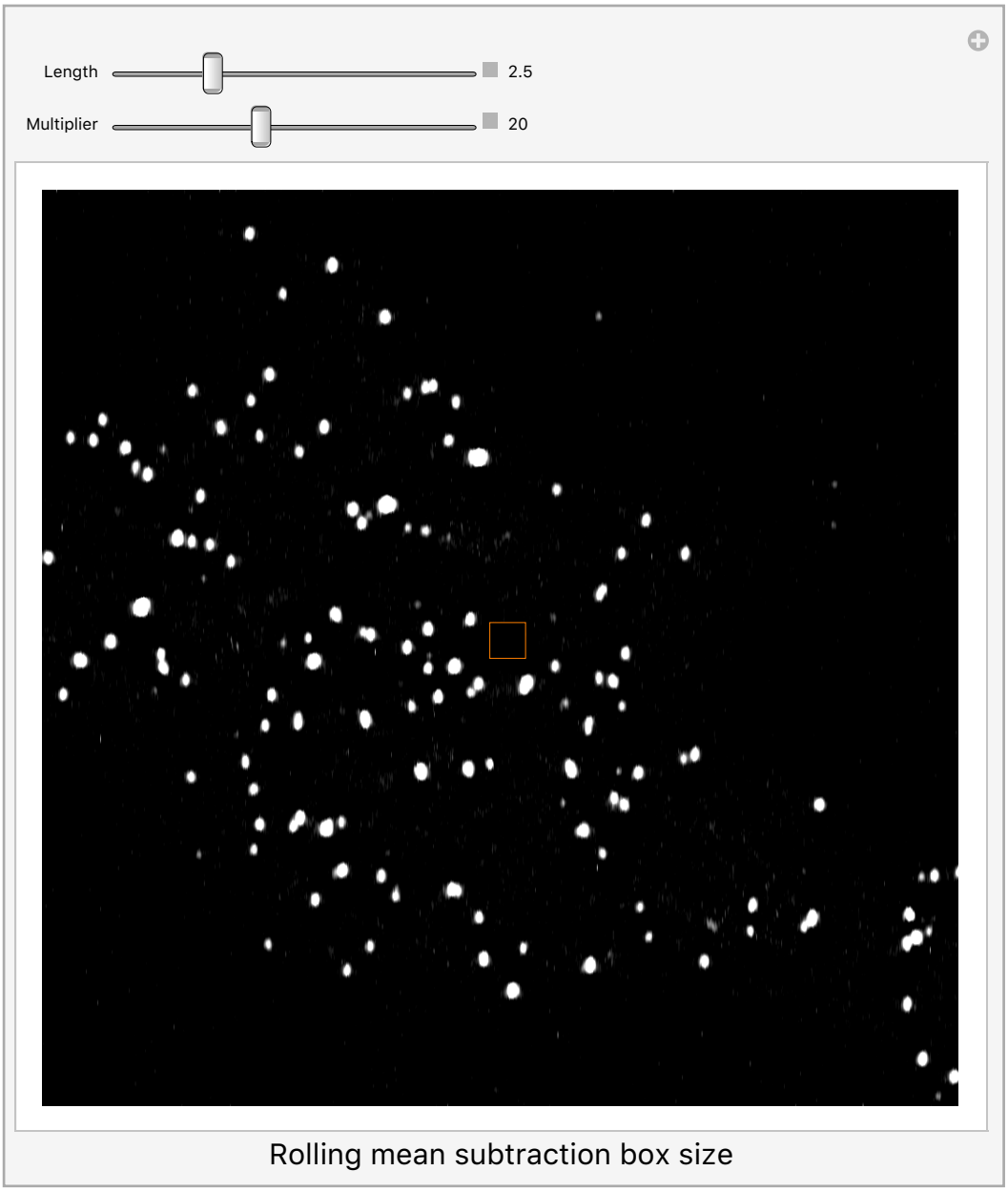
Implementation

Step 2: Local background subtraction

Choose the dimensions of a rolling mean subtraction box. The box should be larger than the average dot diameter but smaller than the scale of the fluctuation in the background. The location of this box as depicted has no meaning.

In[85]:= **ChooseRollingBox**

Out[85]=



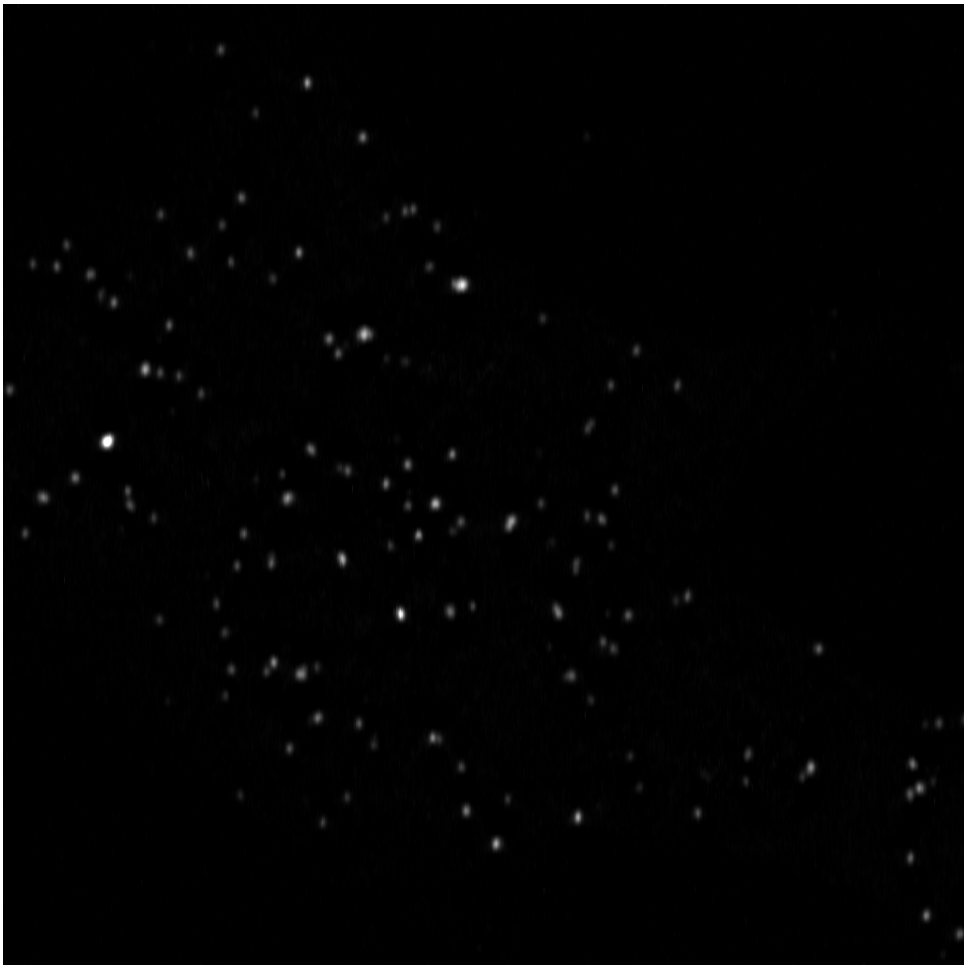
Rolling mean subtraction box size

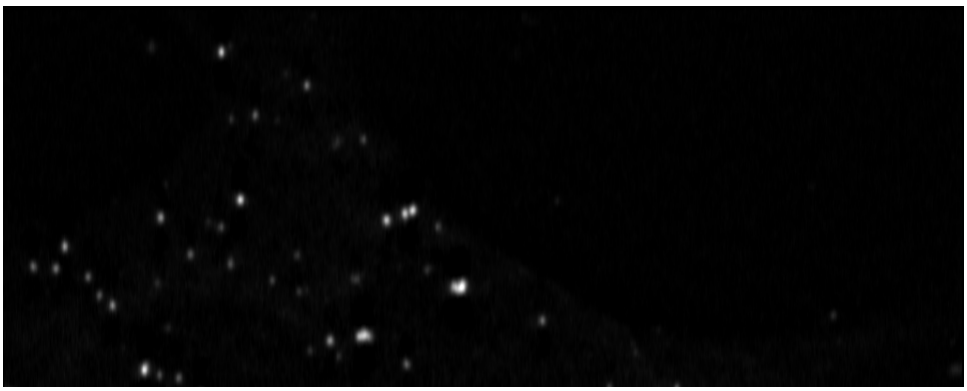
In[86]:=
```
Print[
  AbsoluteTiming[Parallelize[IMG = MapThread[MeanSubtract, {IMG, Round[# / BOX] & /@
          MEANLENGTH, MEANSCALE}];]][[1]], " seconds to subtract means"];
Column[{Text[Style["Images with local background subtraction (channels 1 and 2)",
    20, Bold]], Row[Show[Proj[#, 5], ImageSize → 500] & /@ IMG, Spacer[1]]}]
```

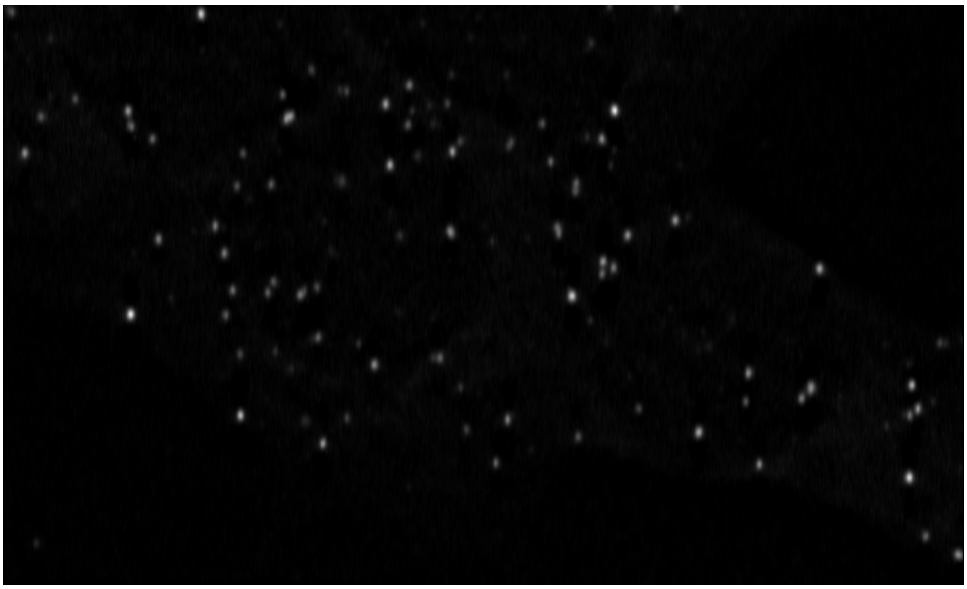
```
9.83268 seconds to subtract means
```

Images with local background subtraction (channels 1 and 2)

Out[87]=





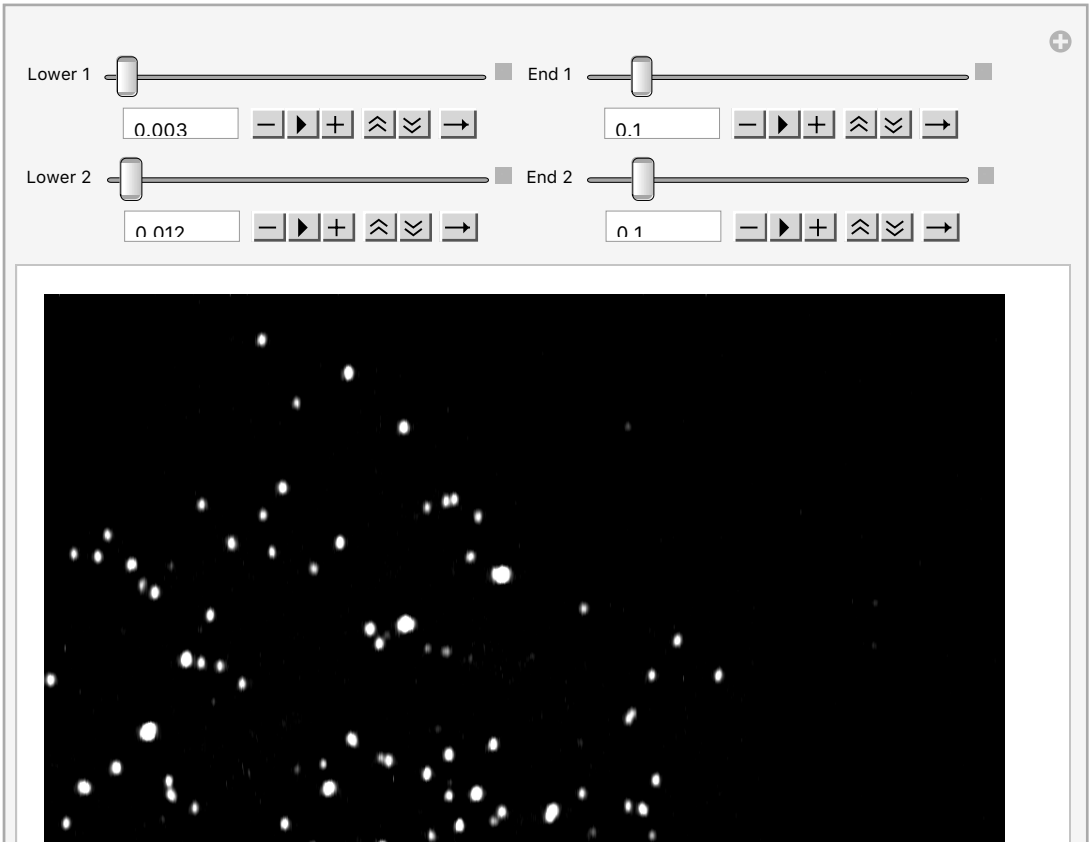


Implementation

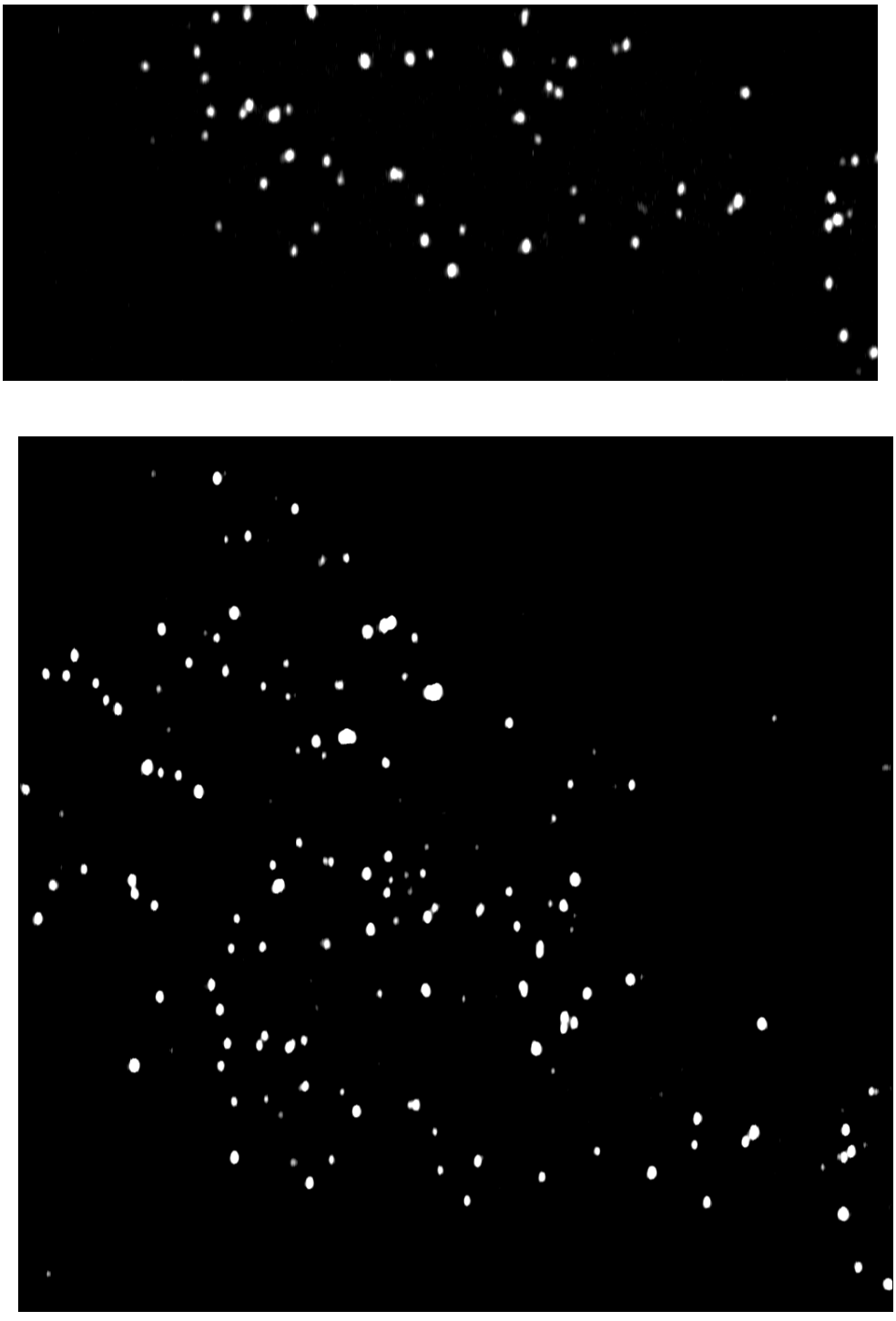
Step 3: Global threshold on pixel intensity

Next we chop off all pixel intensities below a given threshold in each channel.

In[89]:= **ChooseBrightnessThresholds**



Out[89]=



Set the global lower thresholds for the images

In[90]:=
```
Print[
  AbsoluteTiming[Parallelize[IMG = MapThread[LinearShift, {IMG, LOWER, UPPER}];]][[
   1]], " seconds to apply thresholds"];
```

```
0.491733 seconds to apply thresholds
```

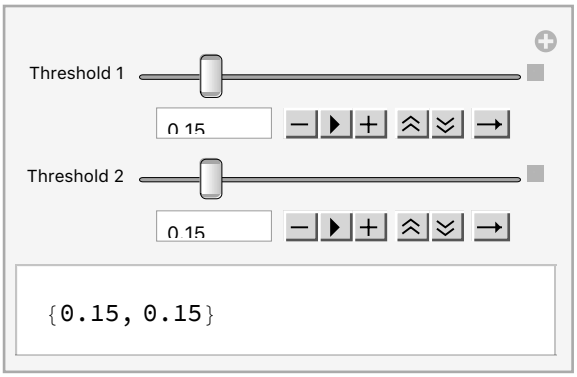
Implementation

Step 4: Watershed dot detection

The watershed method is generally too slow for interactive use, so we pick the thresholds beforehand. If the results are unsatisfactory, we go back, change the thresholds, and rerun the method.

In[94]:=
```
Manipulate[WATERSHED = {w1, w2},
 {{w1, WATERSHED[[1]], "Threshold 1"}, 0, 1, Appearance → "Open"},
 {{w2, WATERSHED[[2]], "Threshold 2"}, 0, 1, Appearance → "Open"}]
```

Out[94]=

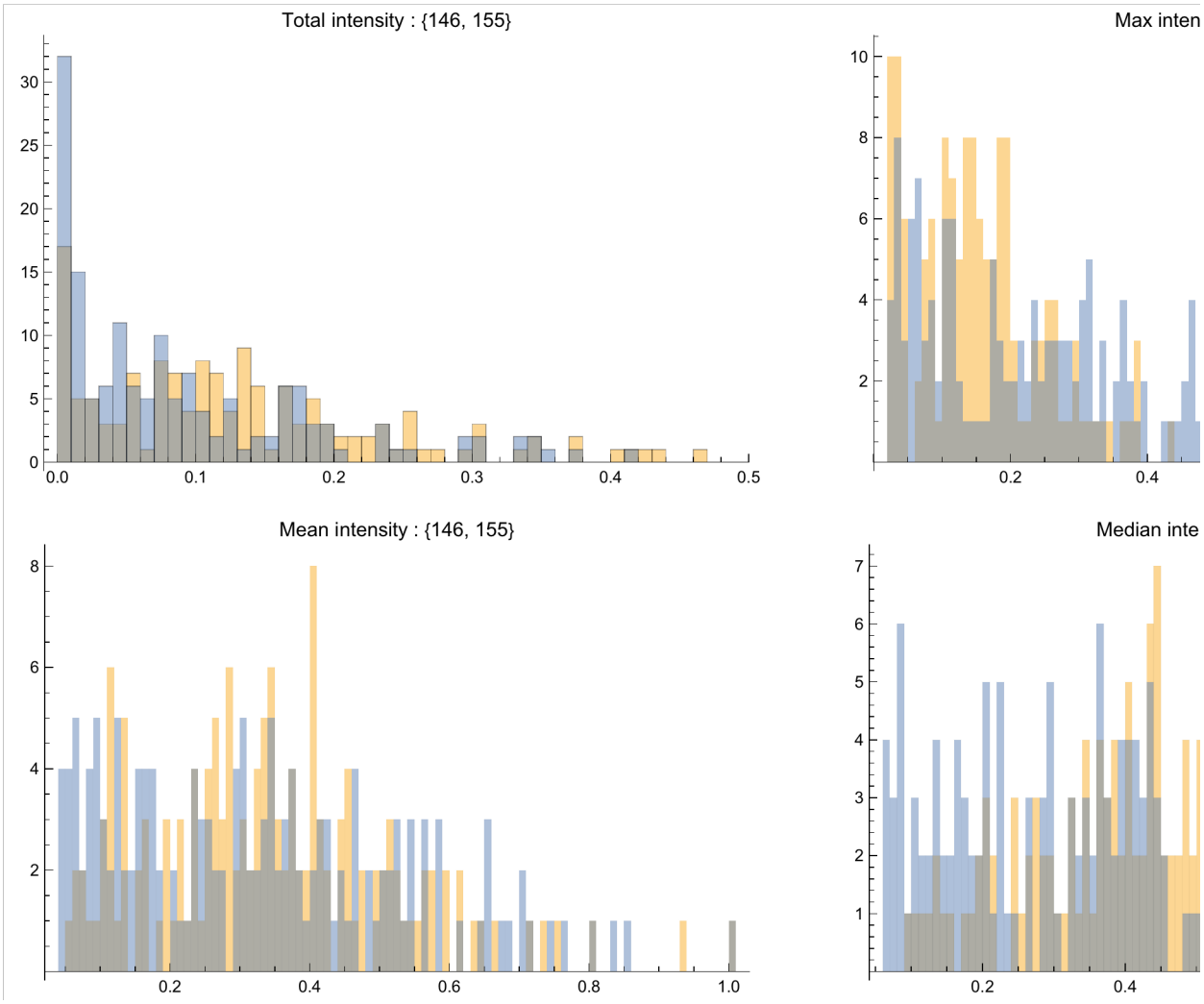


Display the results of the dot segmentation analyses by watershed method in both channels. The number of total dots detected are in the titles of the plots, and both channels are depicted simultaneously (channel 1=blue, channel 2=yellow).

In[95]:= **RunWatershed[]**

```
14.2838 seconds to apply watershed method
```

Out[95]=

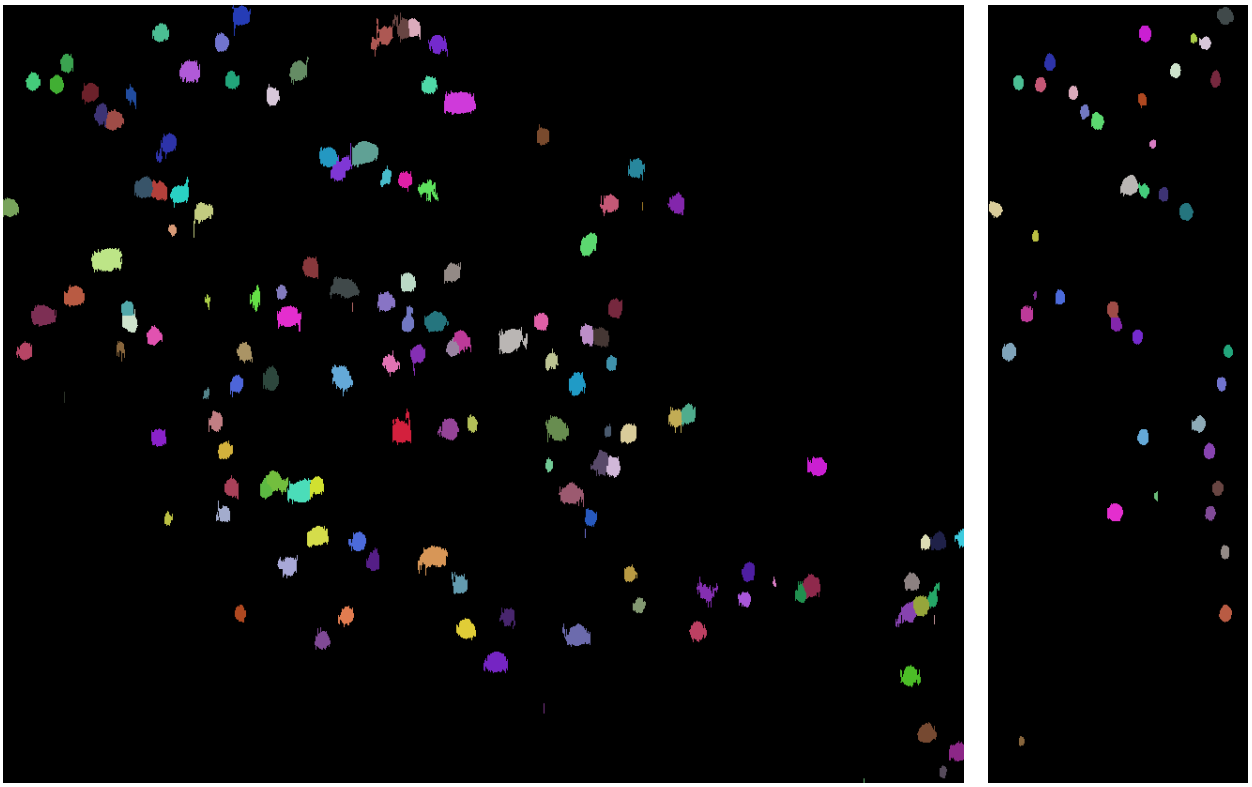


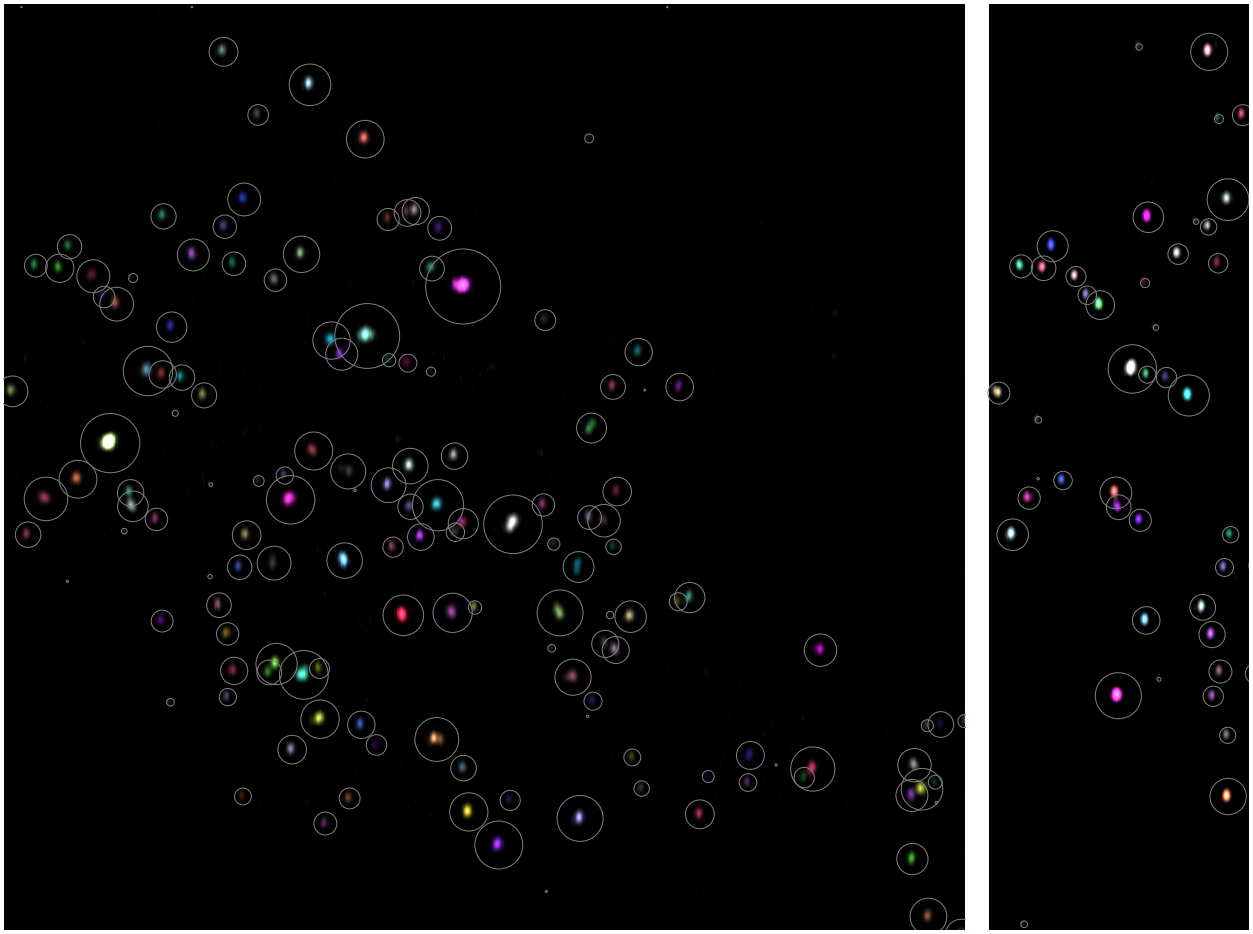
To get some sense of where the watershed method is segmenting vs. agglomerating dots, we display images of the deduced dots labeled by their index.

In[96]:= **WatershedDisplay[]**

Top left: watershed basins in channel 1
Top right: watershed basins in channel 2
Bottom left: circled watershed centroids in channel 1
Bottom right: circled watershed centroids in channel 2

Out[96]=



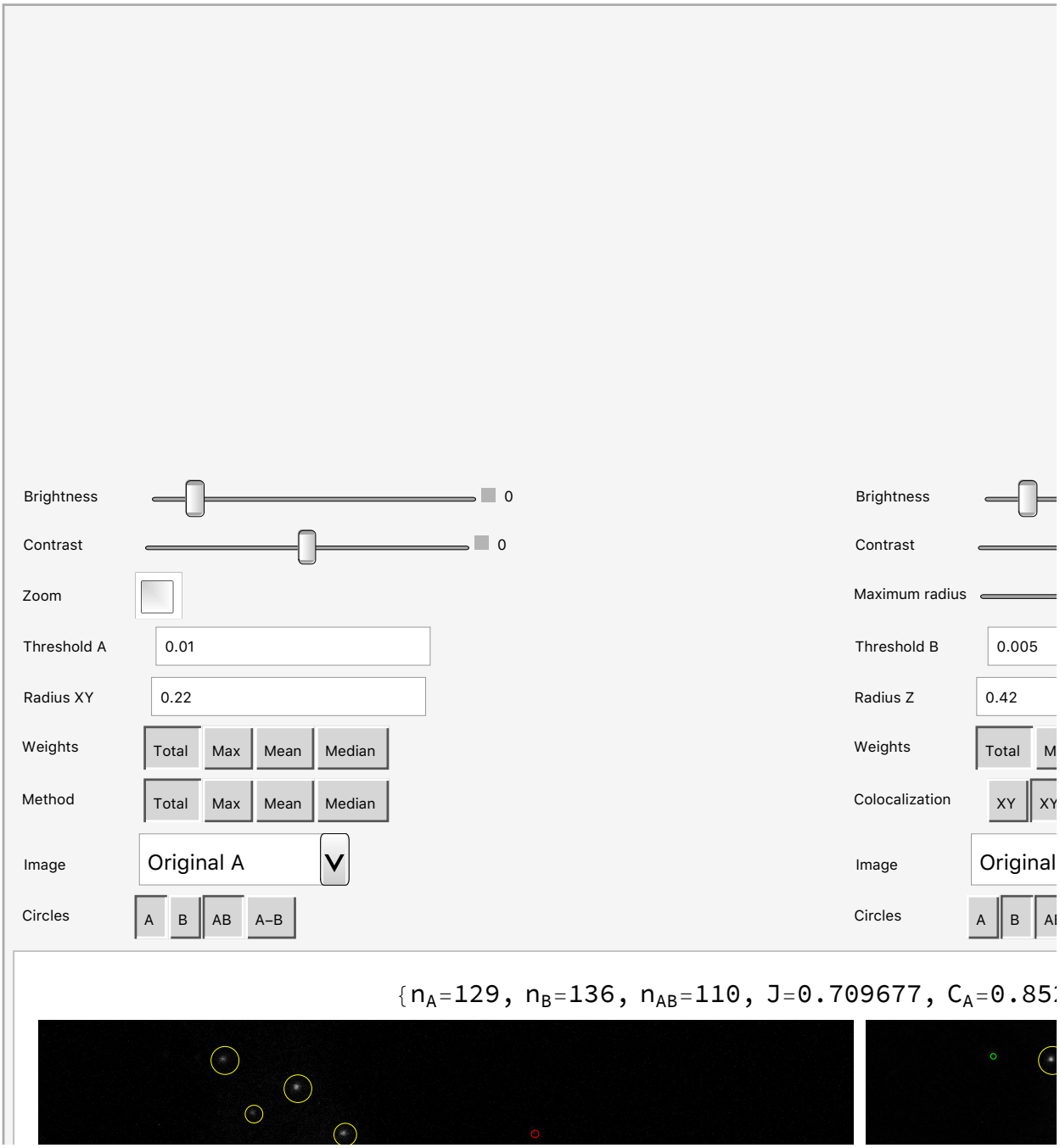


Implementation

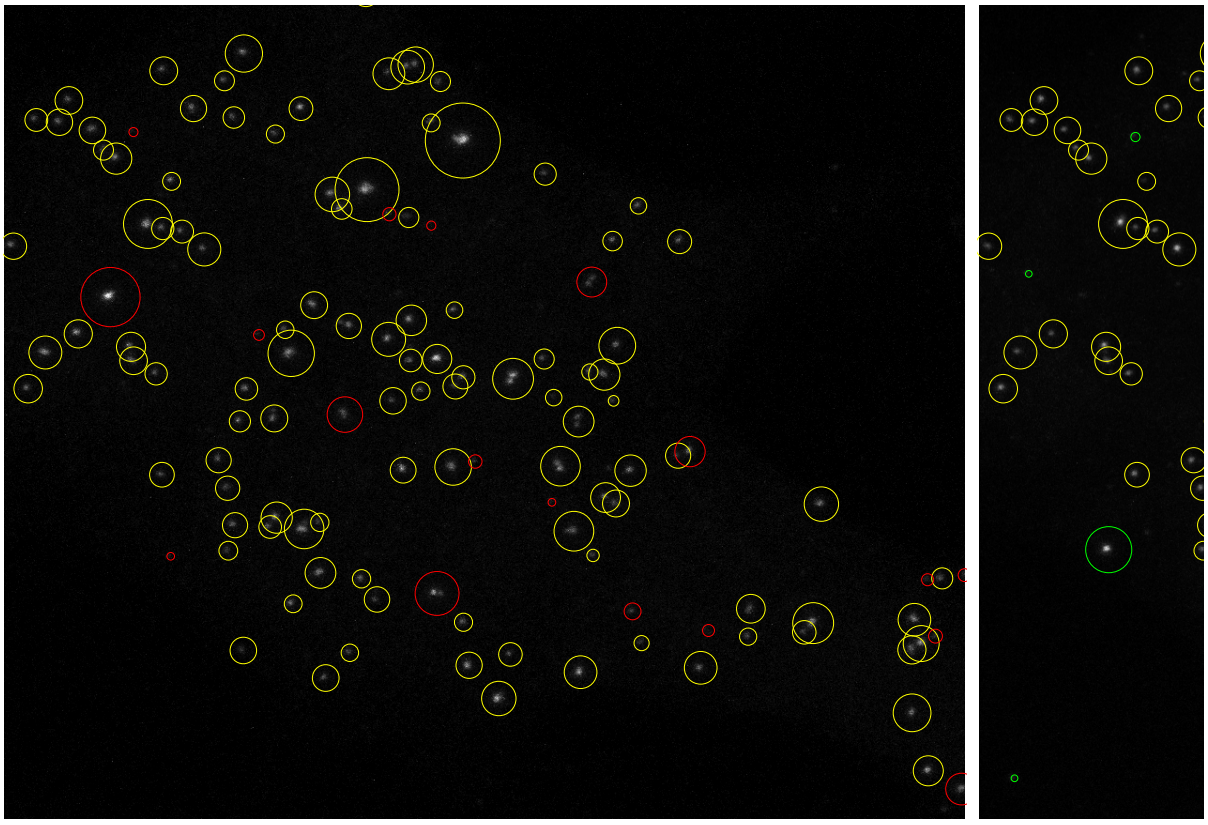
Step 5: Global threshold on dot intensity

In this section we run interactive selection of dot intensity thresholds in channels A and B. Once the thresholds are set as desired, go to the next cell and export results.

In[100]:= `ThresholdGUI`



Out[100]=



Implementation

Final image output

In this section we generate final publication quality graphics of the colocalized dots with scale bars. Image cropping and rearrangement of scale bars may be done later in Adobe Illustrator or a similar program. EPS files will be generated, so be sure to re-click the button if parameters are changed!

In[103]:=
```
ExportButton[]
```

Out[103]=

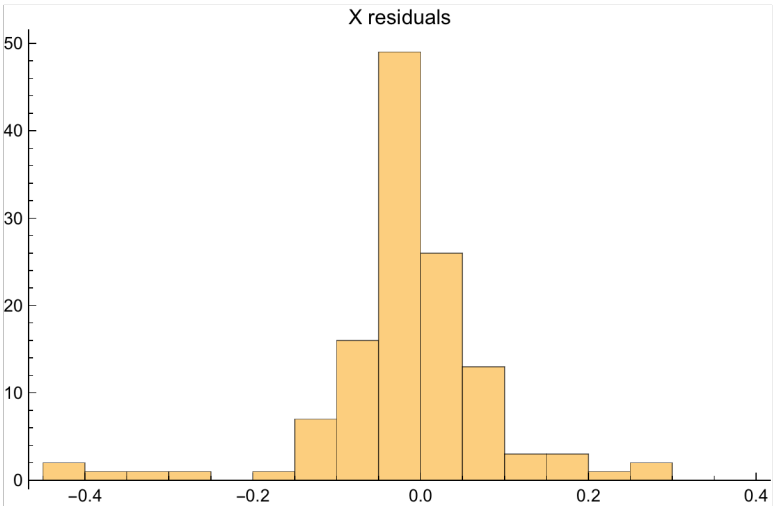
Export final EPS images and JSON

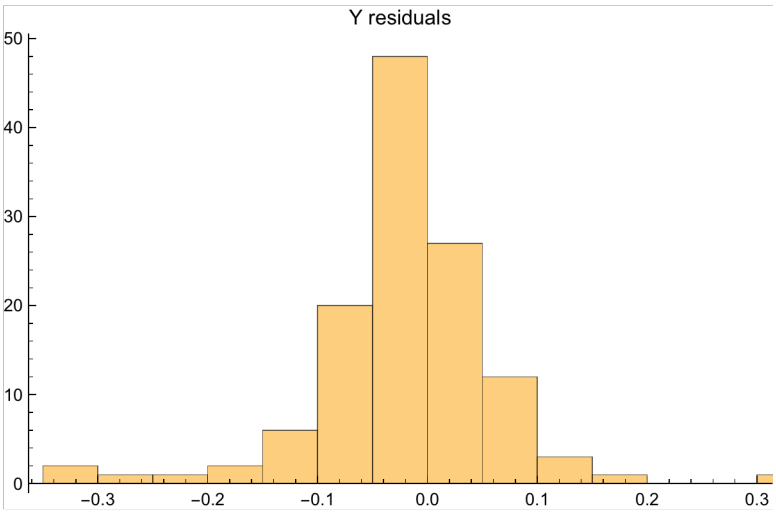
Implementation

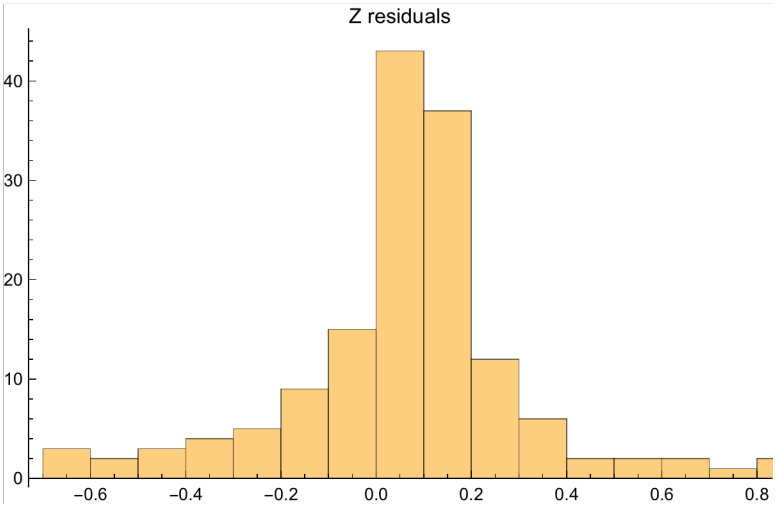
Histograms

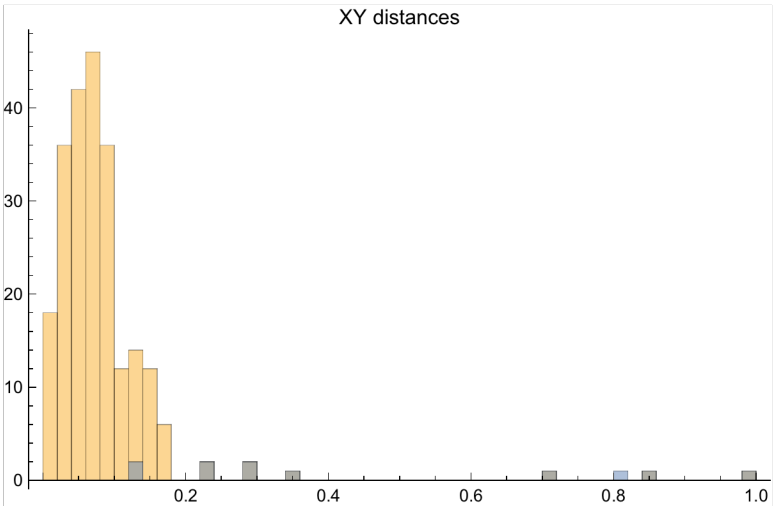
In this section we examine the residual distances between dots detected in channel A and dots detected in channel B via histograms

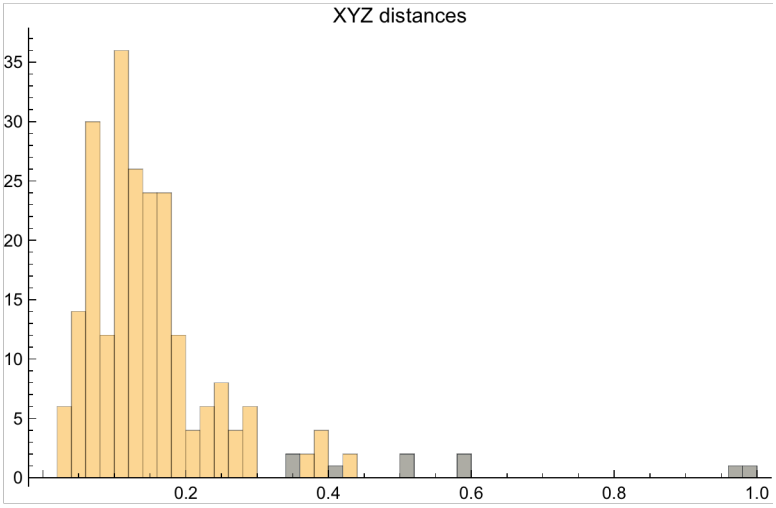
In[105]:=
```
ShowHistograms[]
```


X residuals
50
40
30
20
10
0
−0.4
−0.2
0.0
0.2
0.4


Y residuals
50
40
30
20
10
0
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3


Z residuals
40
30
20
10
0
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8


XY distances
40
30
20
10
0
0.2
0.4
0.6
0.8
1.0


XYZ distances
35
30
25
20
15
10
5
0
0.2
0.4
0.6
0.8
1.0