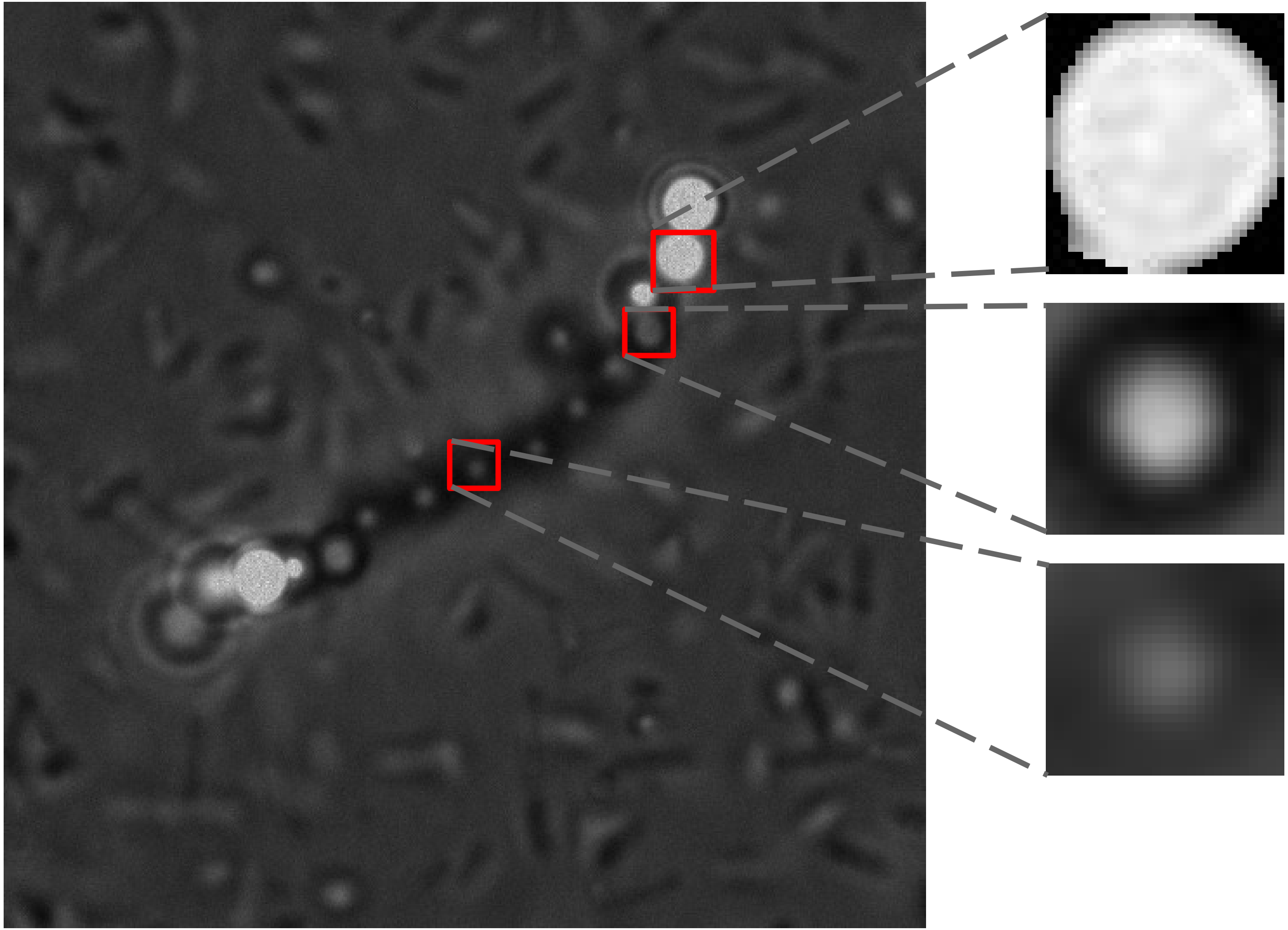
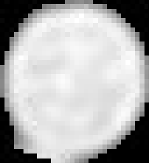
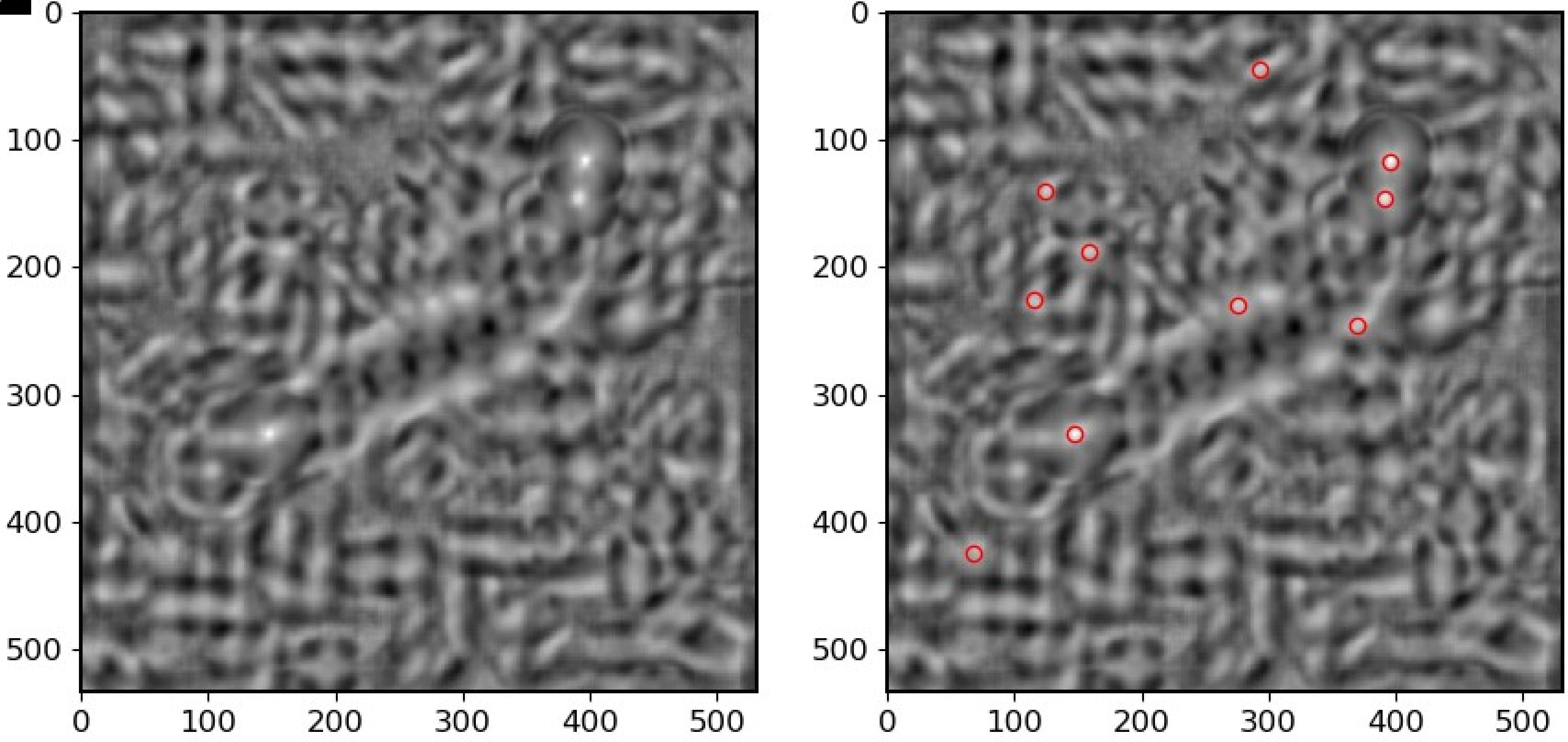
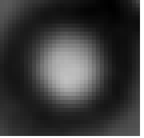
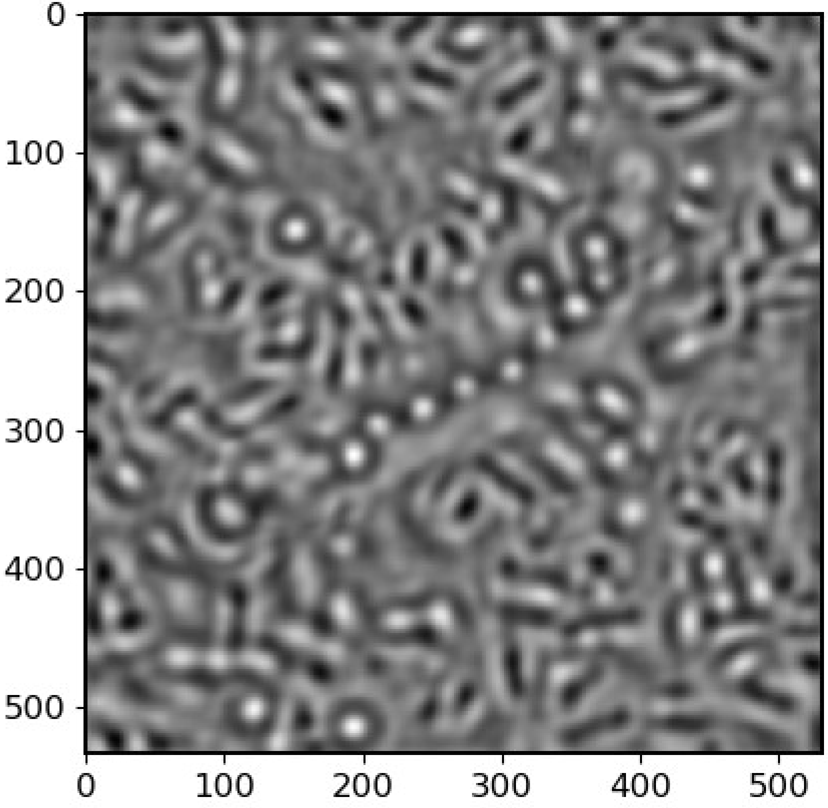


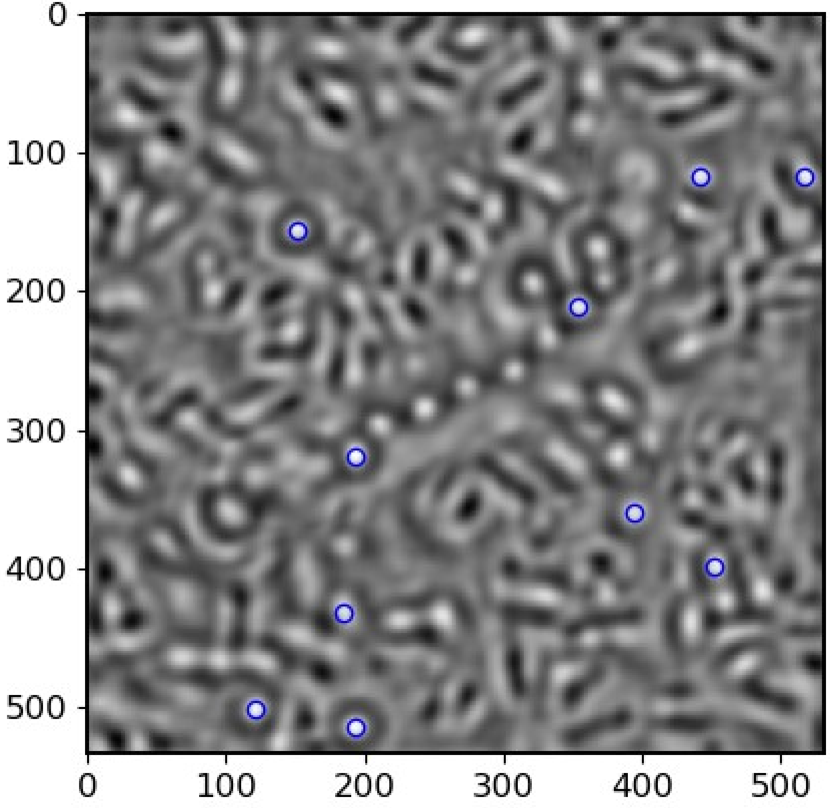


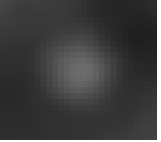


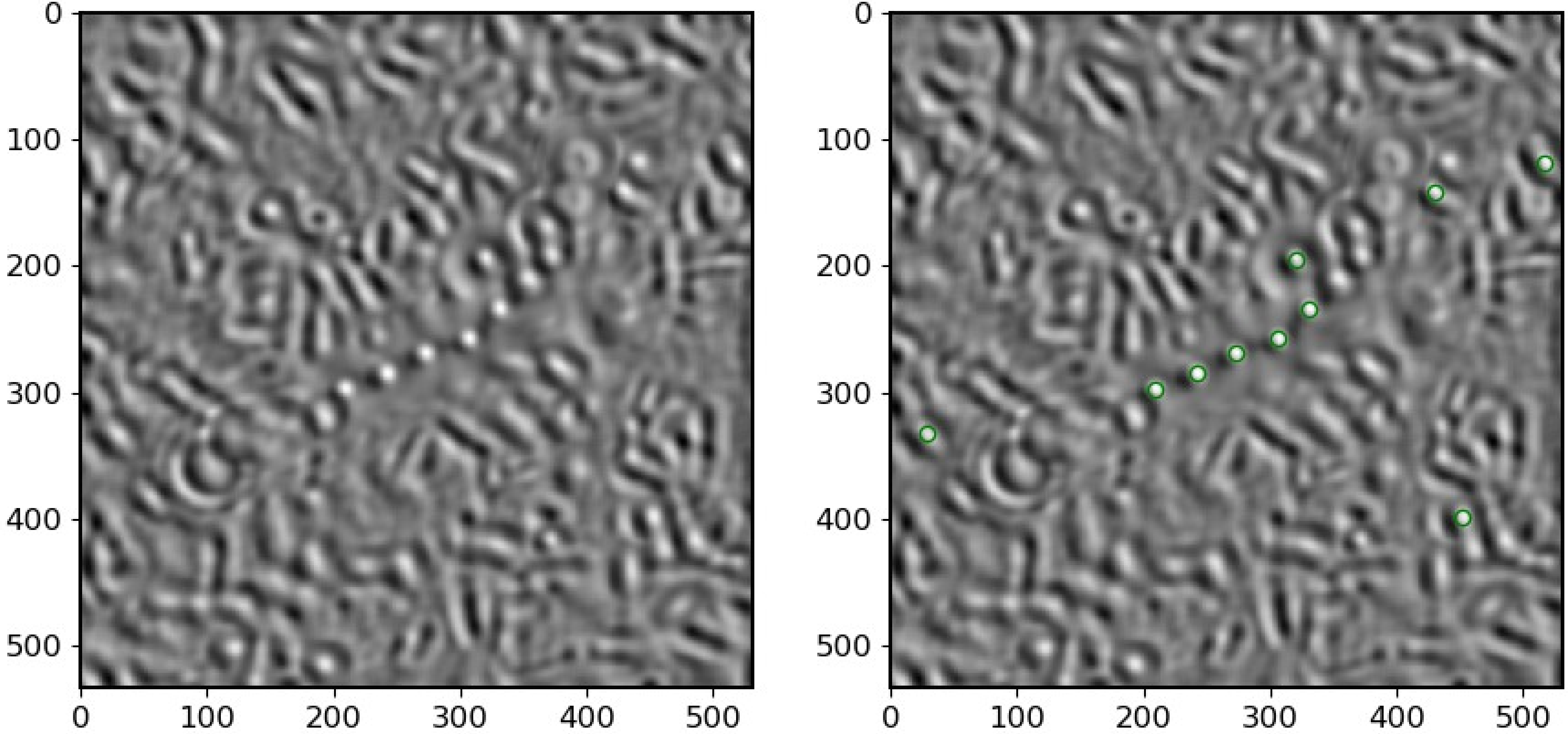
0
100
200
300
400
500
0
100
200
300
400
500

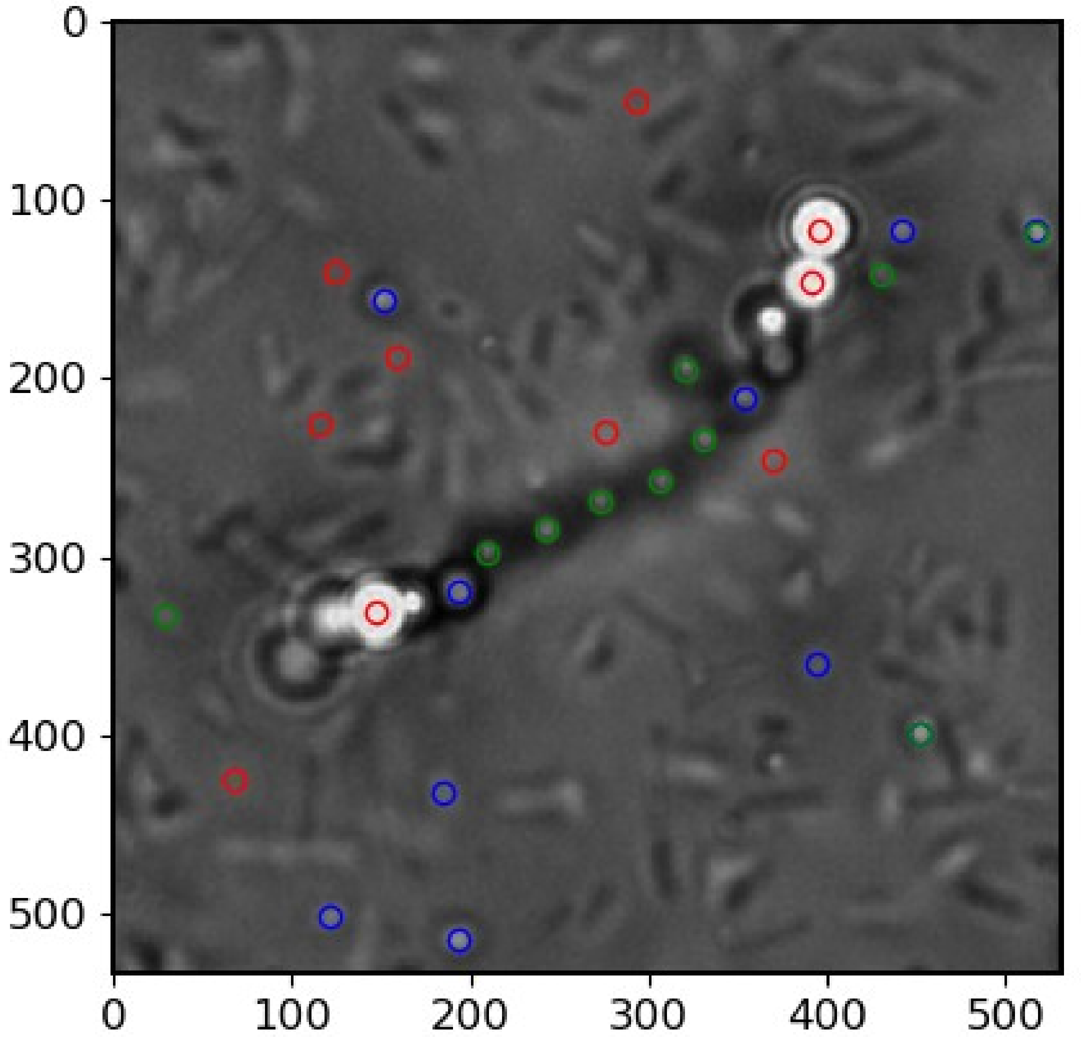



0
100
200
300
400
500
0
100
200
300
400
500

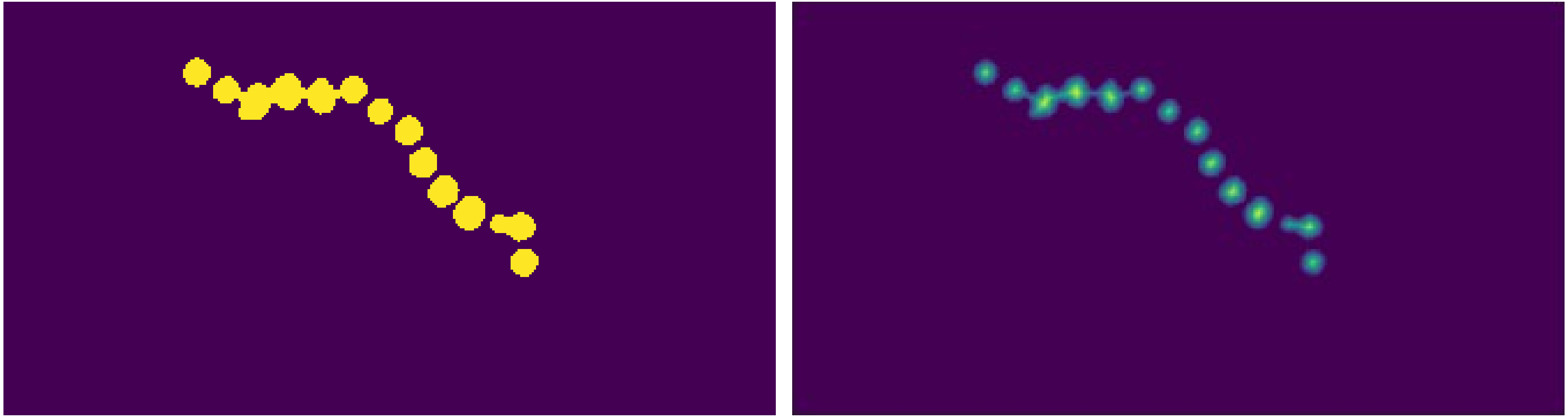

0
100
200
300
400
500
0
100
200
300
400
500




0
100
200
300
400
500
0
100
200
300
400
500
0
100
200
300
400
500
0
100
200
300
400
500

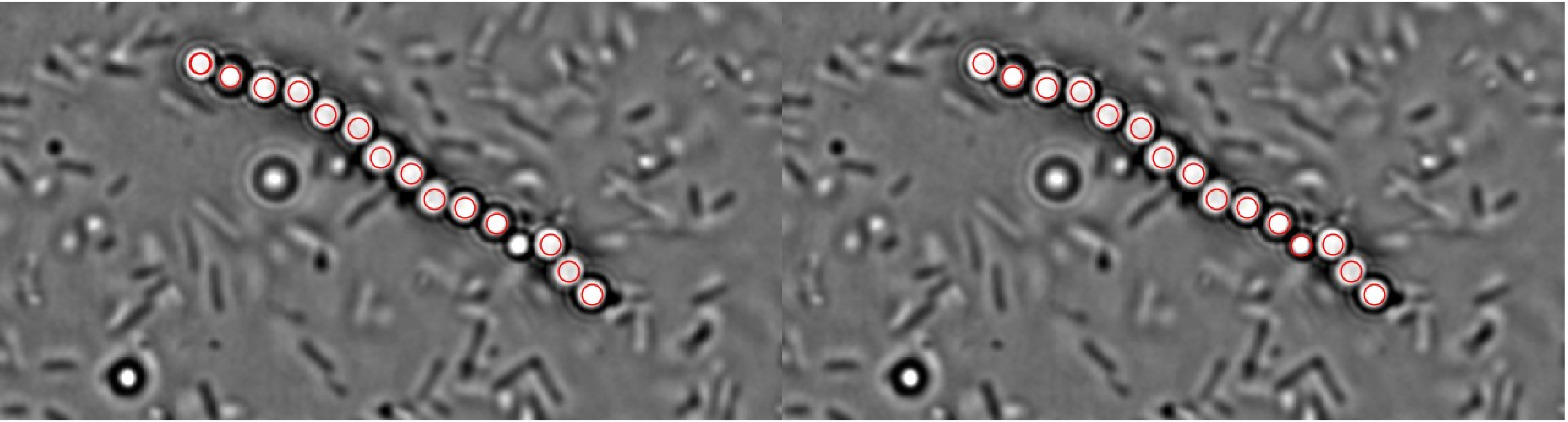

0
100
200
300
400
500
0
100
200
300
400
500

Distance transformation



Min distance check

Min_dist = 0　　　Min_dist = 10



Peak score

- Assign scores to each peak based on the likelihood that the peak has an corresponding particle
- Sort peaks according to the score (rather than peak values)

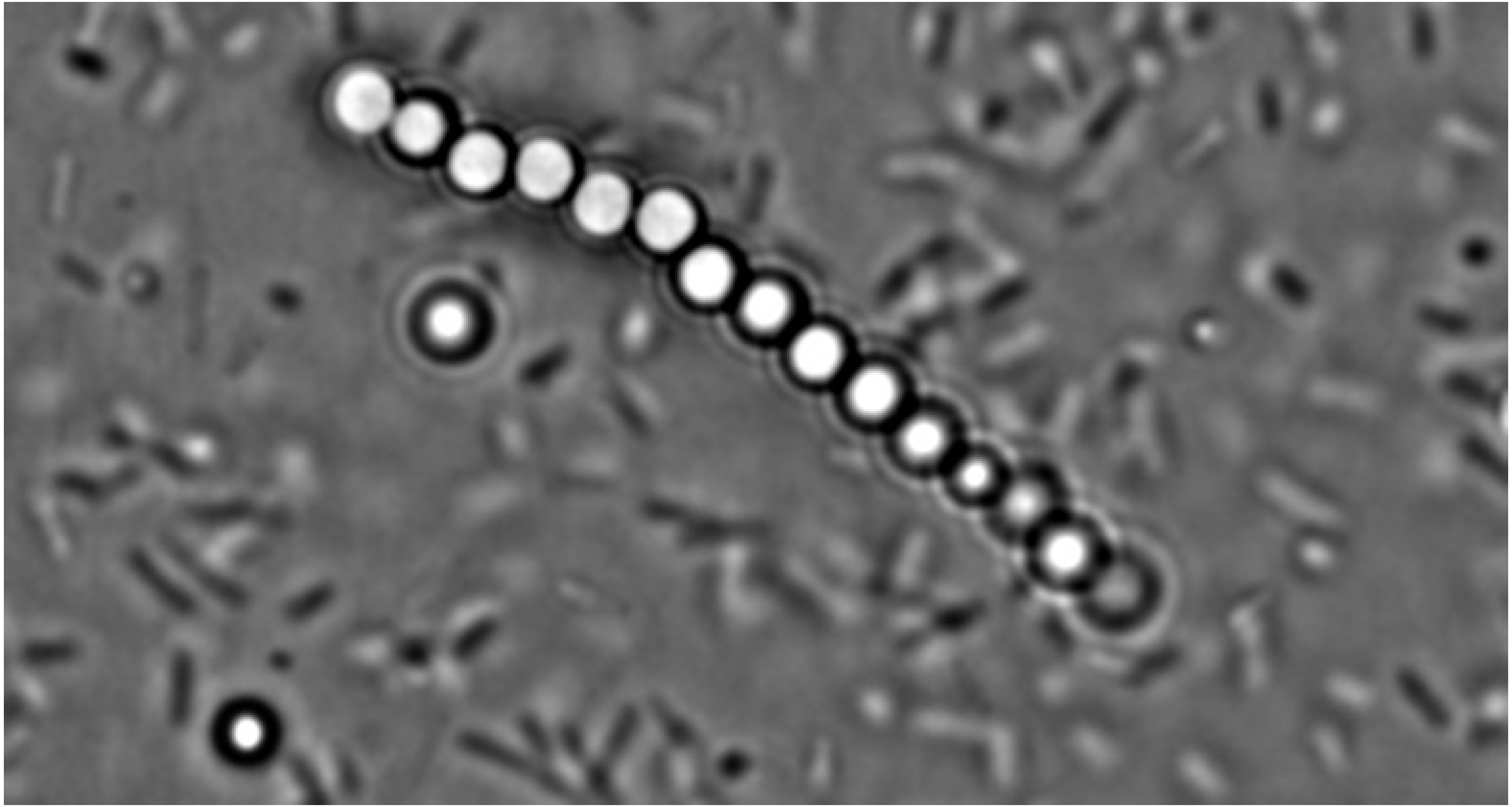
Colloidal chain tracking framework

- Preprocessing
- Prelim tracking on dt
- Sorting
  - Total pixel intensity
  - Compare to a mask
- Refine result
  - Distance check
  - Total number of particles
  - Gaussian fitting to get subpixel resolution

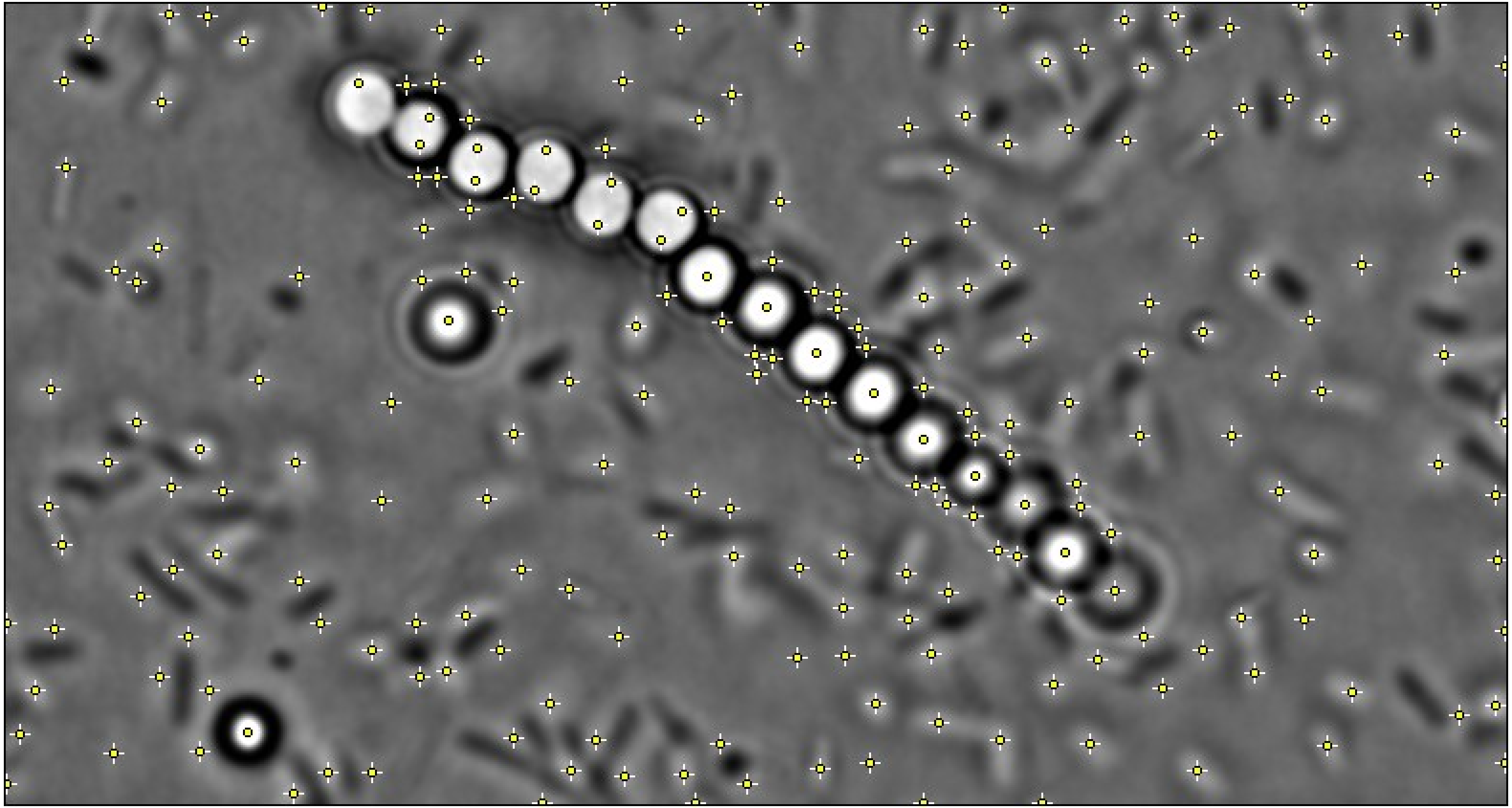
Colloidal chain tracking framework

- Preprocessing
- Prelim tracking on dt
- Sorting
  - Total pixel intensity
  - Compare to a mask
- Refine result
  - Distance check
  - Total number of particles
  - Gaussian fitting to get subpixel resolution

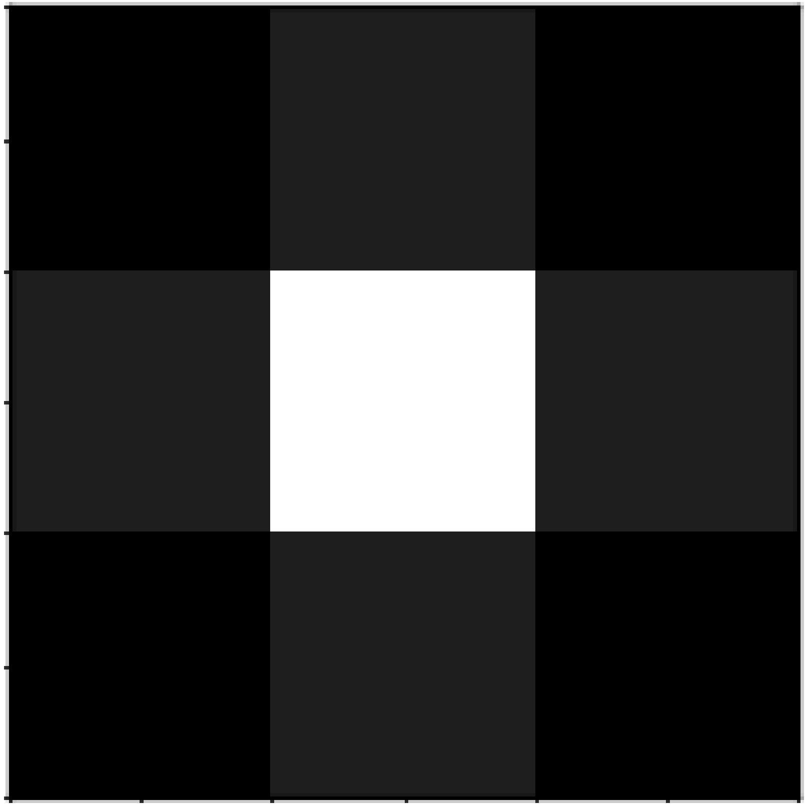
Preprocessing



Preprocessing - find maxima



What is “maxima”?



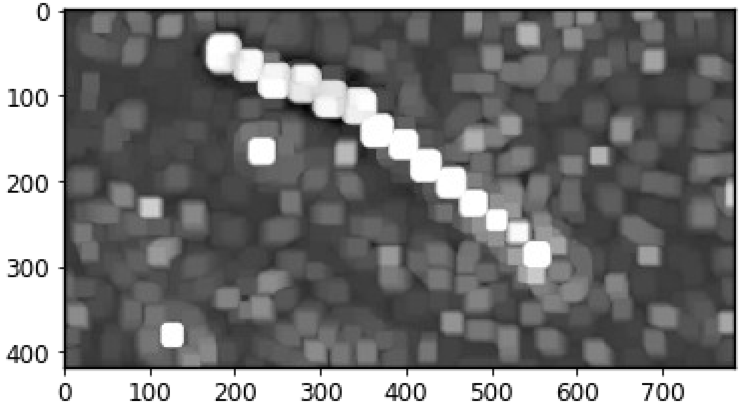
Code

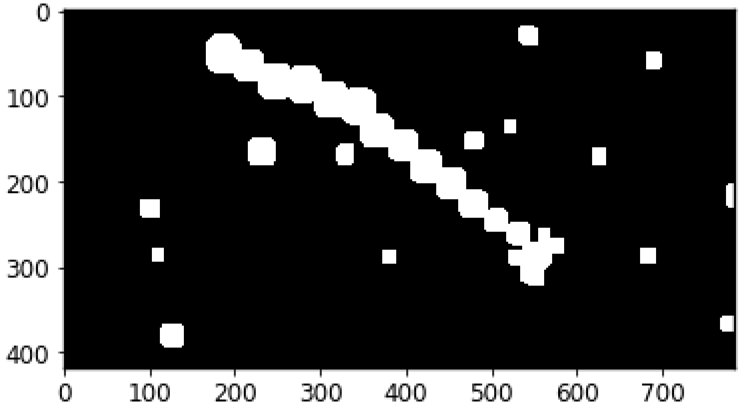
```
if img[x, y] >= img[x-1, y-1] and \
        img[x, y] > img[x-1, y] and \
        img[x, y] >= img[x-1, y+1] and \
        img[x, y] > img[x, y-1] and \
        img[x, y] > img[x, y+1] and \
        img[x, y] >= img[x+1, y-1] and \
        img[x, y] > img[x+1, y] and \
        img[x, y] >= img[x+1, y+1]:
        cent.append(xy)
```

Preprocessing - mask out the chain

Create mask

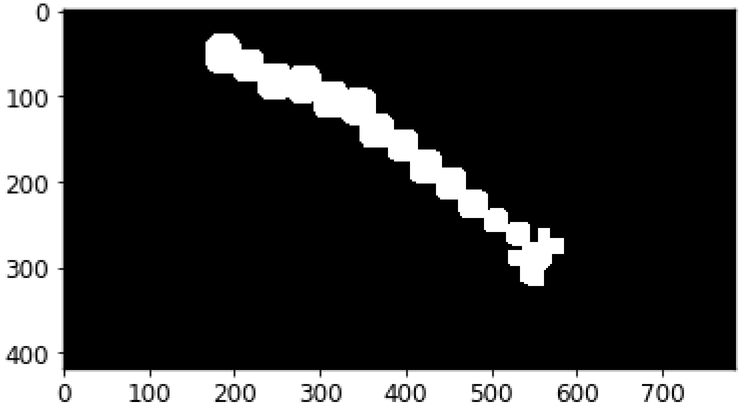
- Maximum filter (size=15)
- Threshold (isodata)
- Make mask (use the biggest connected region)





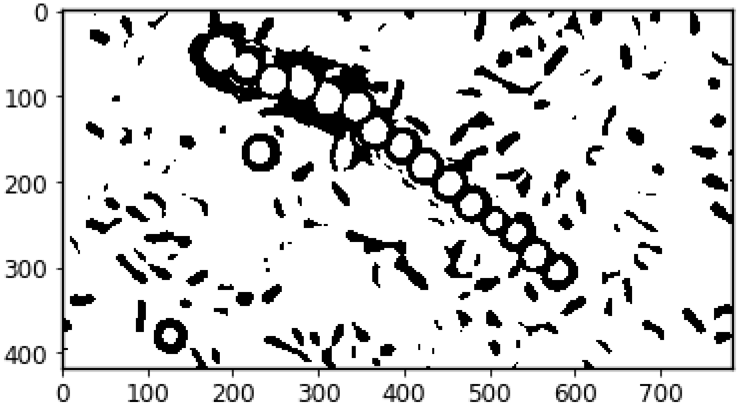
Preprocessing - mask out the chain

Mask



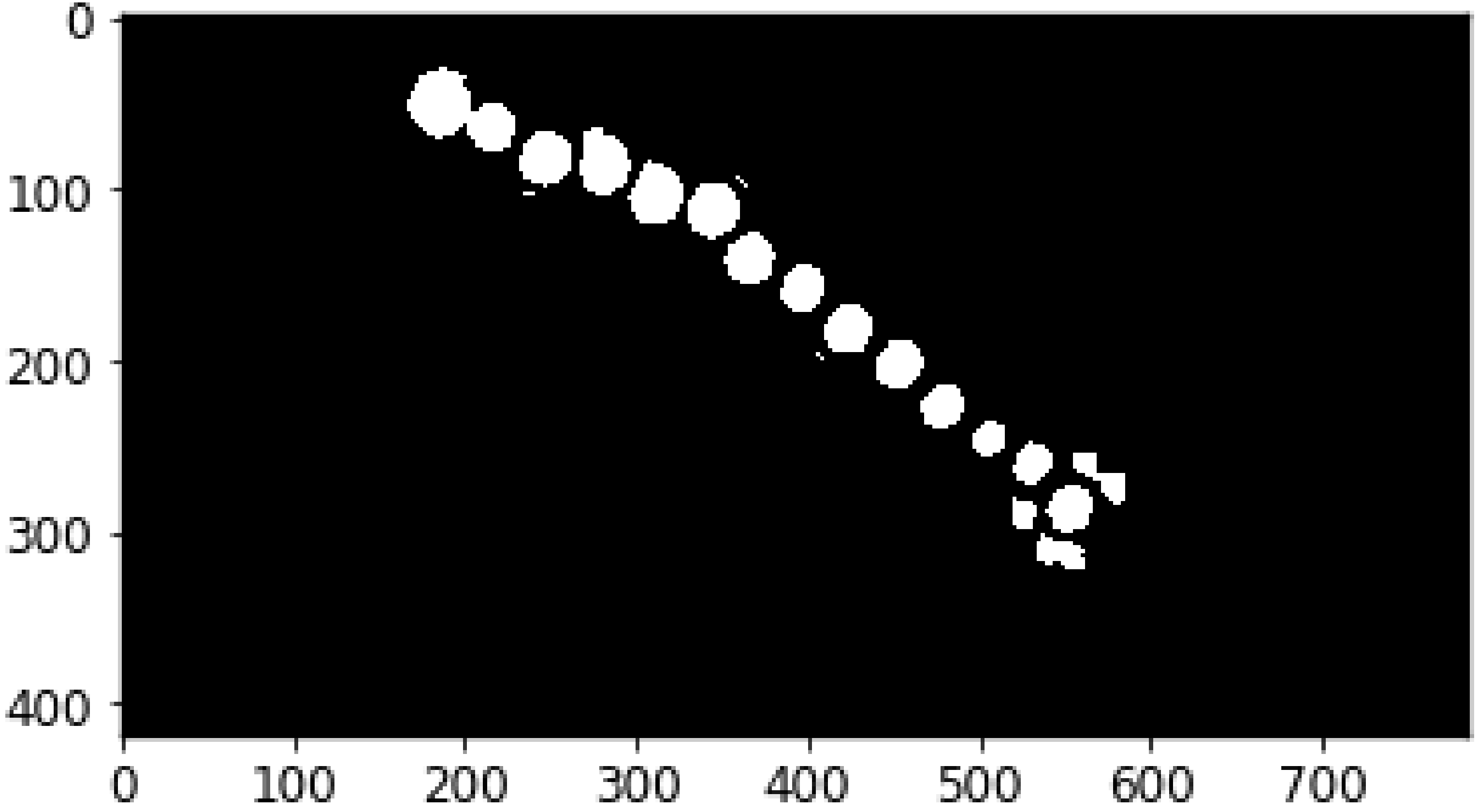
X

threshold

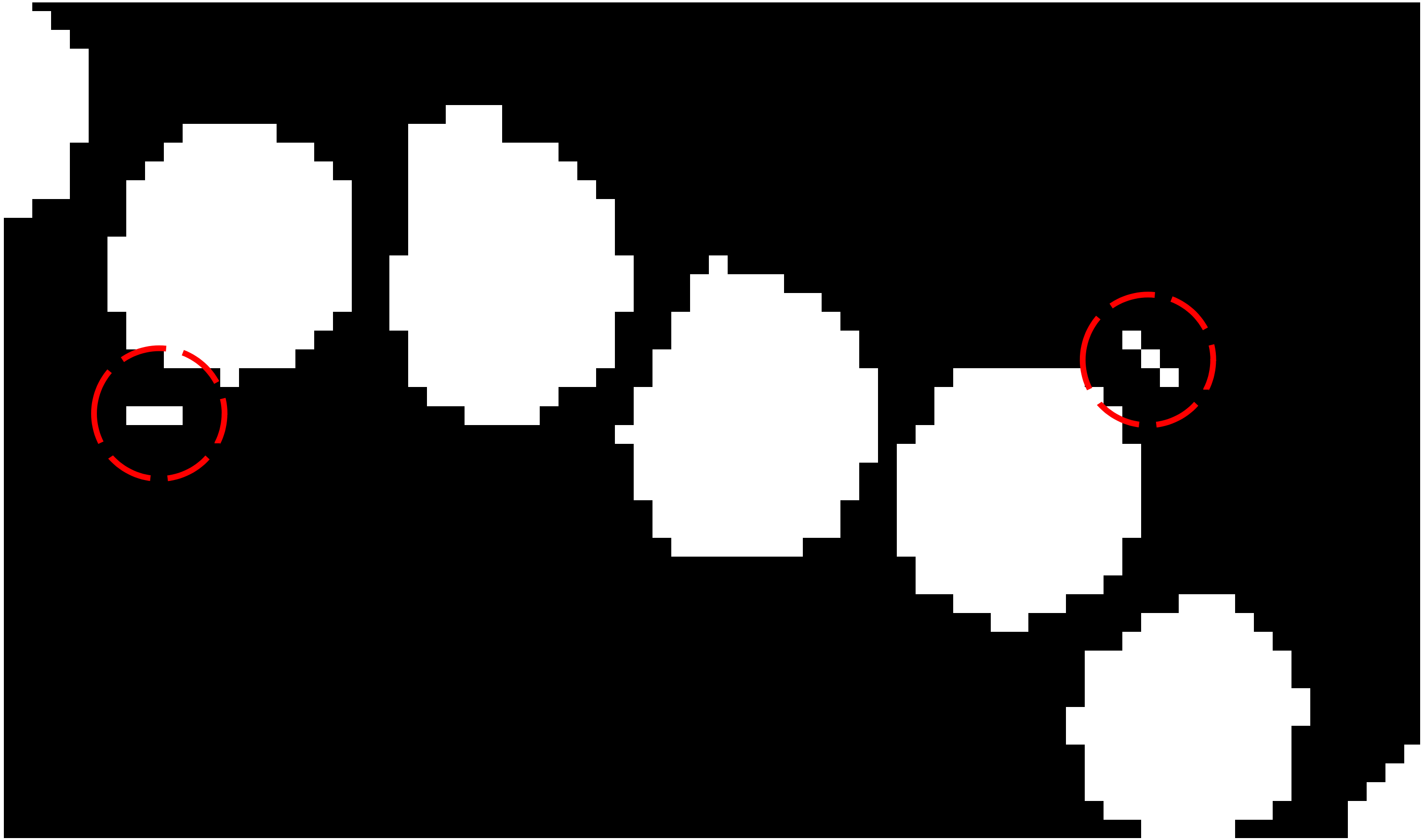


Preprocessing - mask out the chain

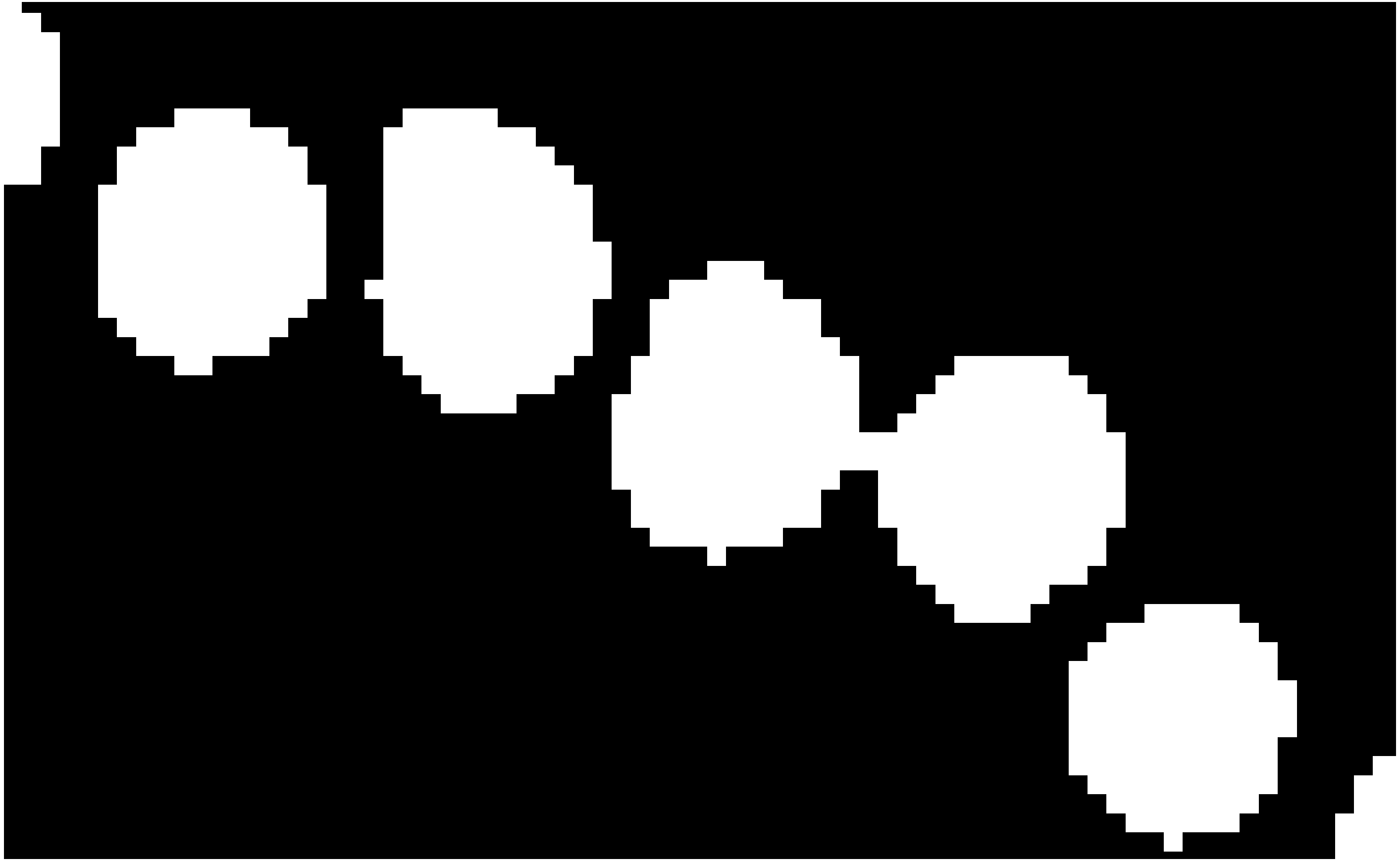
=



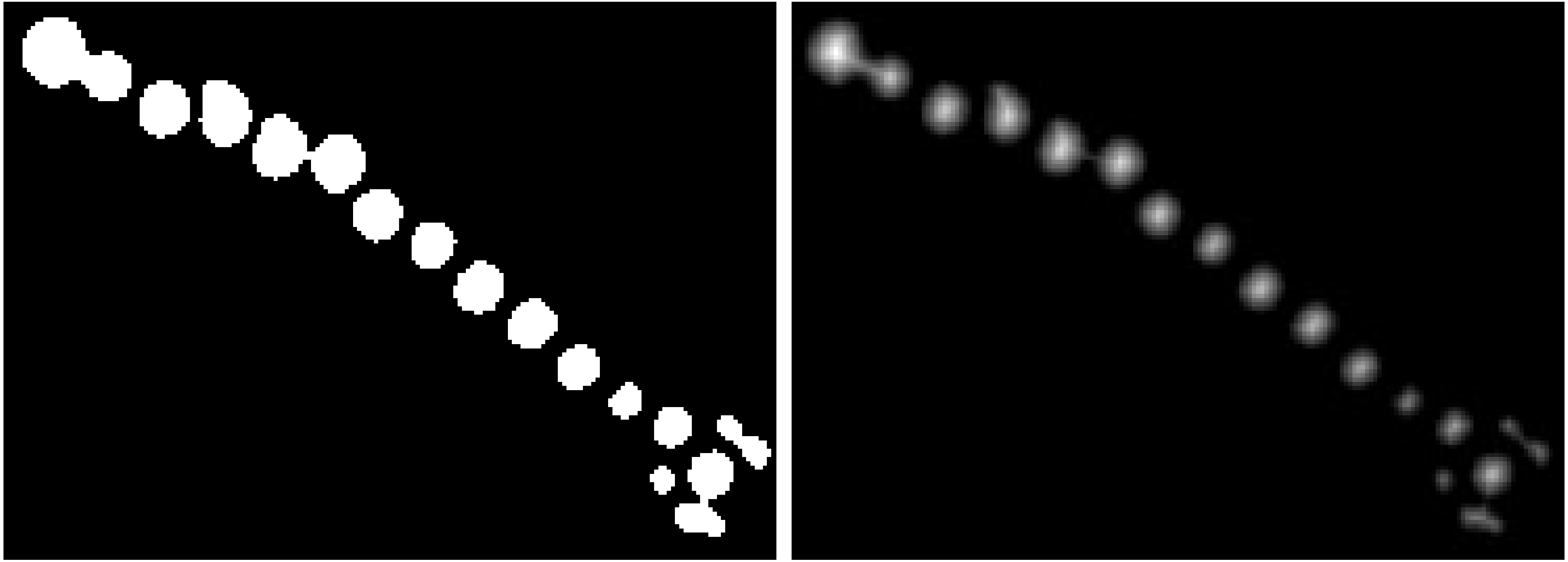
Preprocessing - clear noise (despeckle)



Preprocessing - clear noise (despeckle)



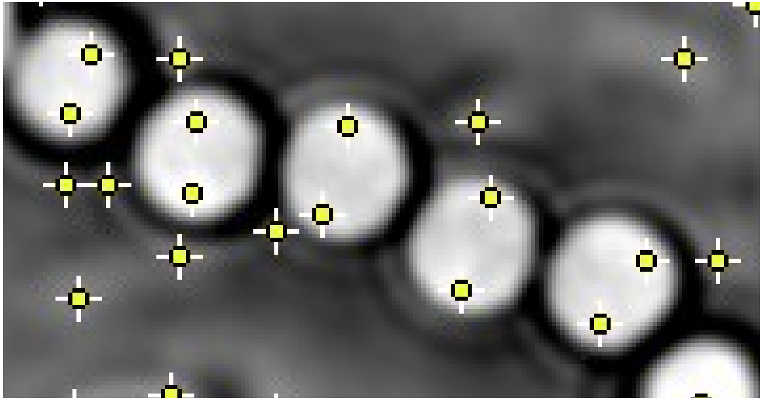
Preprocessing - distance transform (create maxima)

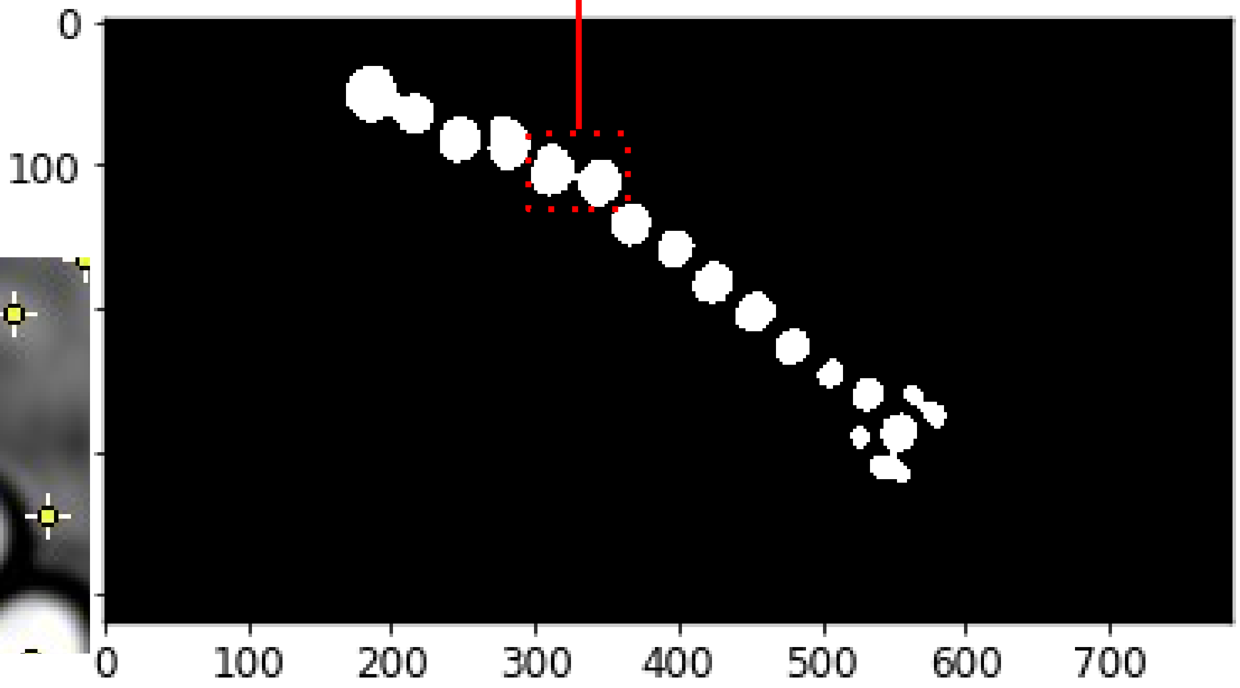


Why is distance transform necessary?

Methods to identify particles

- Connected regions
- Local maxima

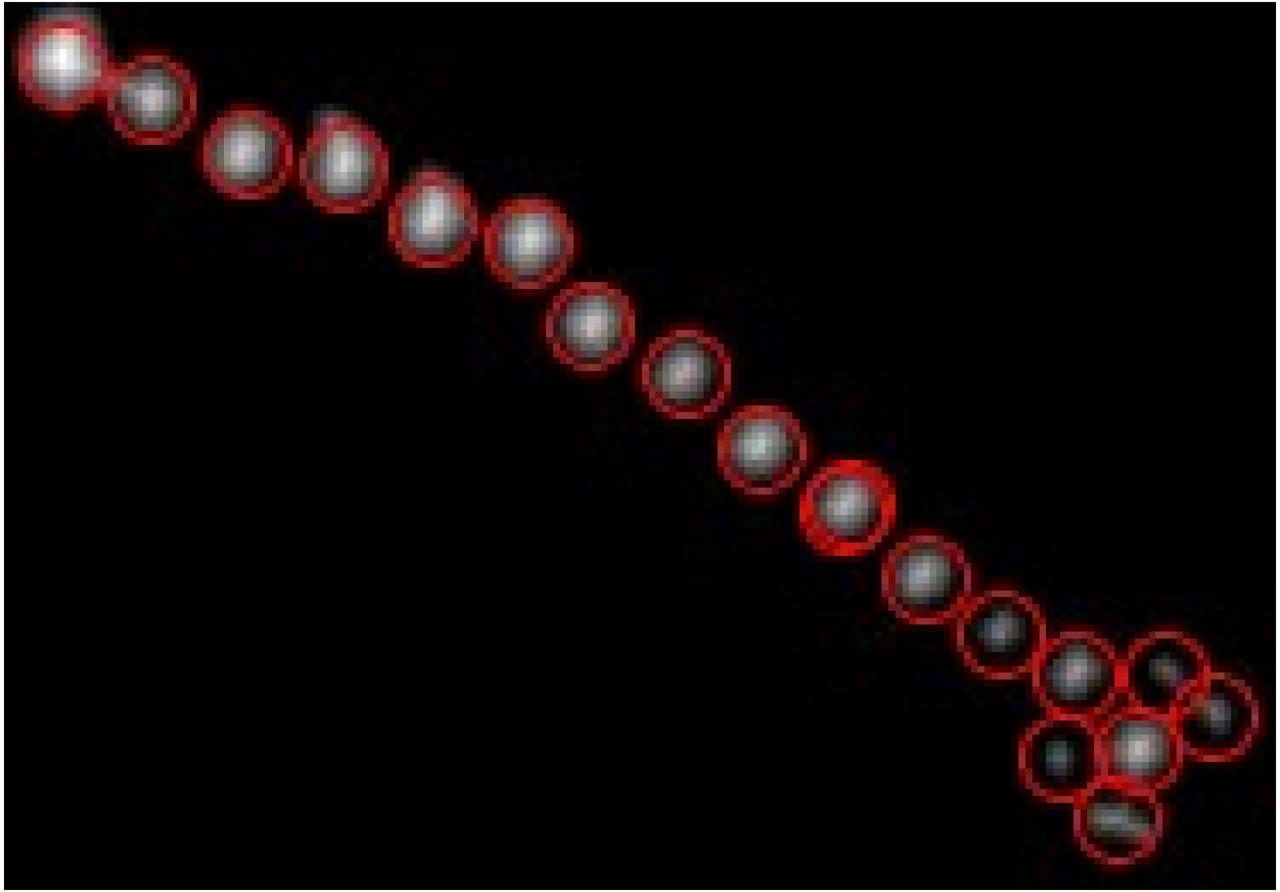




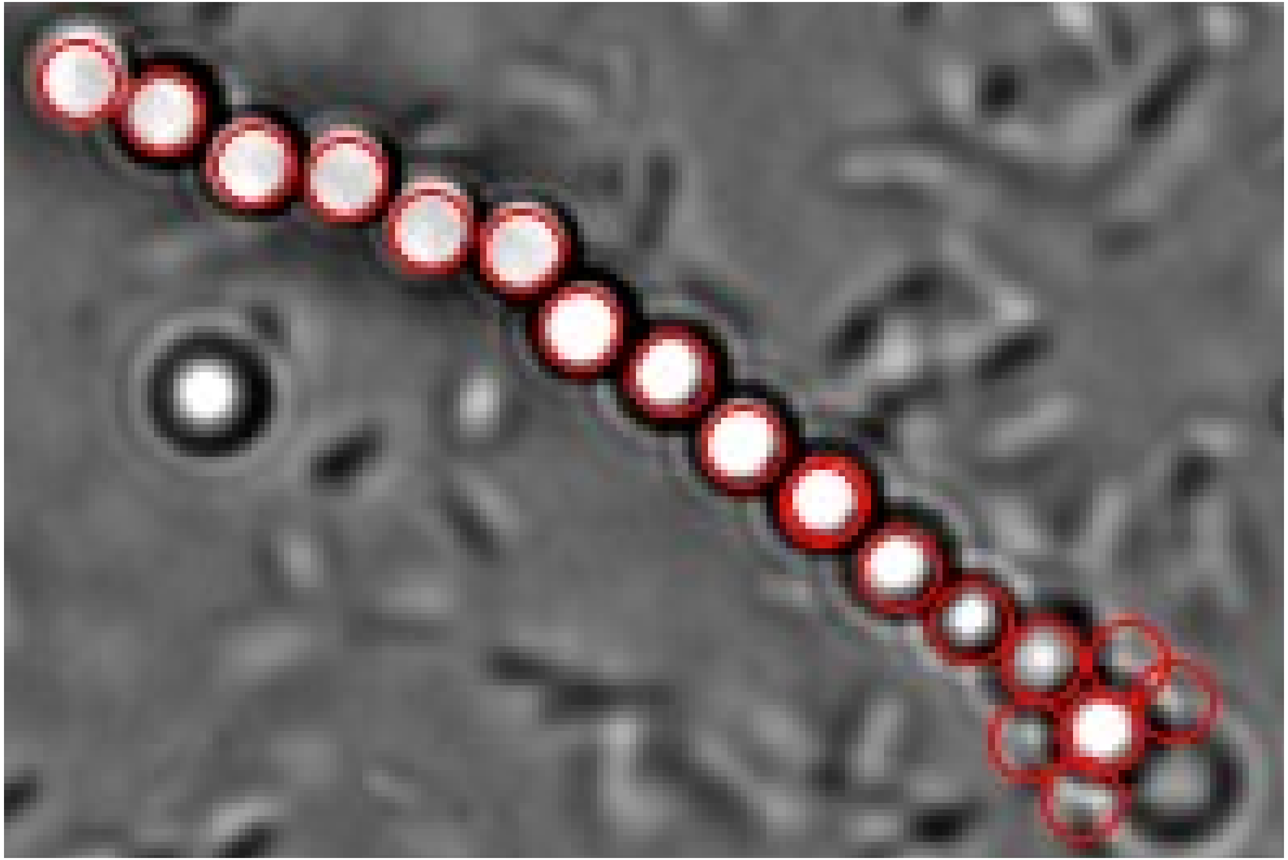
Colloidal chain tracking framework

- Preprocessing
- Prelim tracking on dt
- Sorting
  - Total pixel intensity
  - Compare to a mask
- Refine result
  - Distance check
  - Total number of particles
  - Gaussian fitting to get subpixel resolution

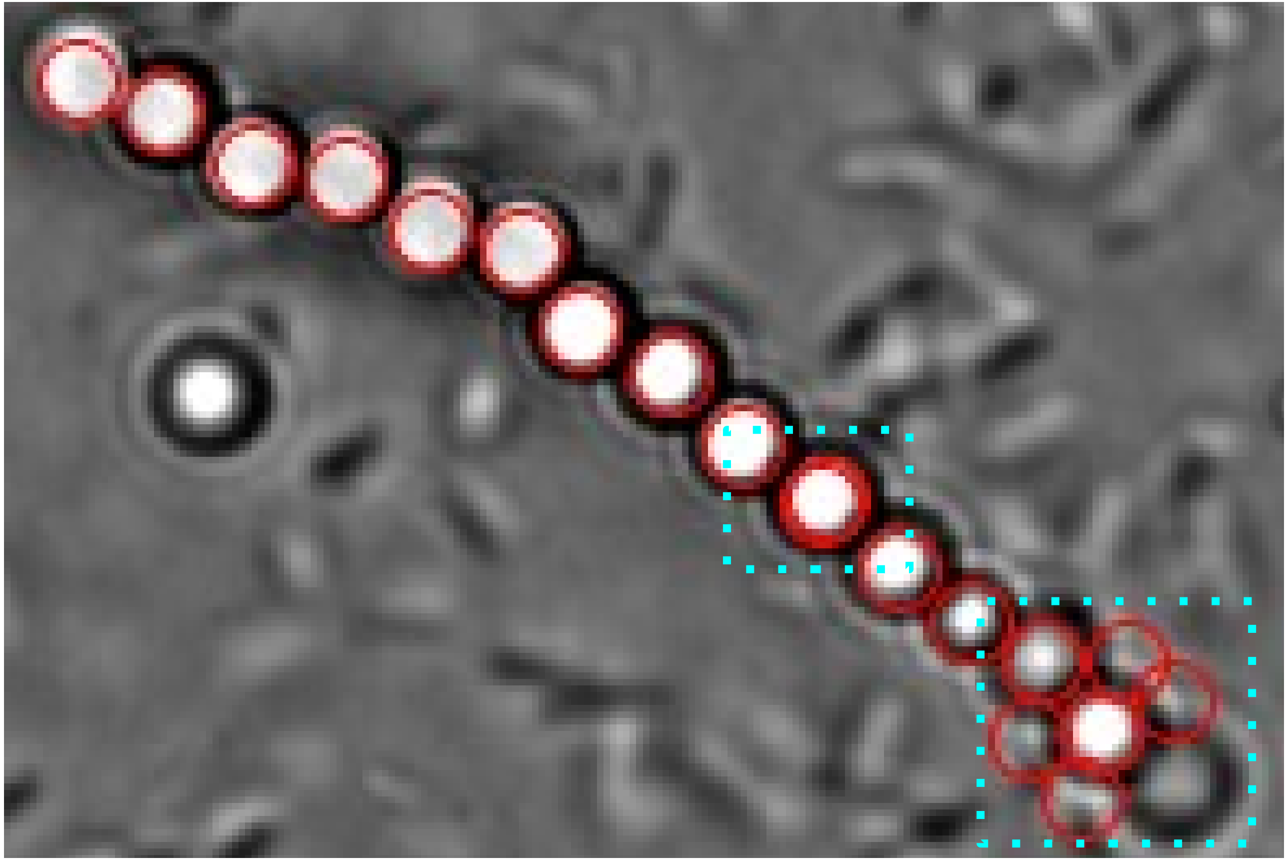
Prelim tracking



Prelim tracking



Prelim tracking

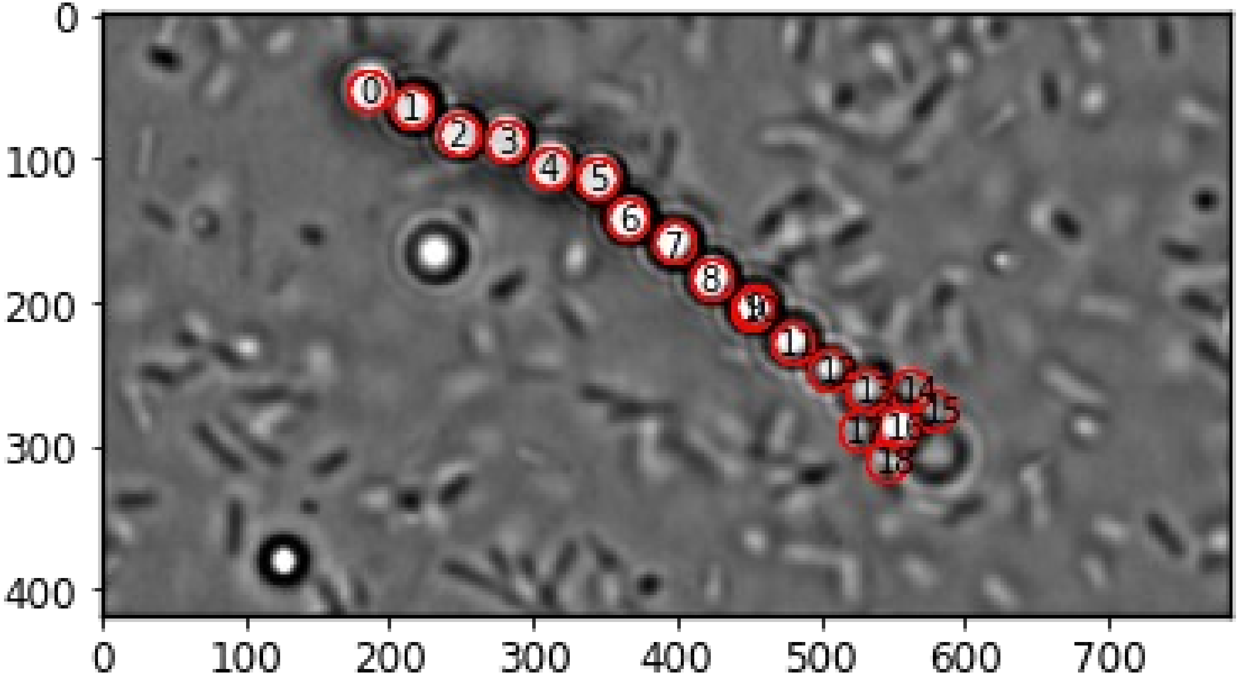


Colloidal chain tracking framework

- Preprocessing
- Prelim tracking on dt
- Sorting
  - Total pixel intensity
  - Compare to a mask
- Refine result
  - Distance check
  - Total number of particles
  - Gaussian fitting to get subpixel resolution

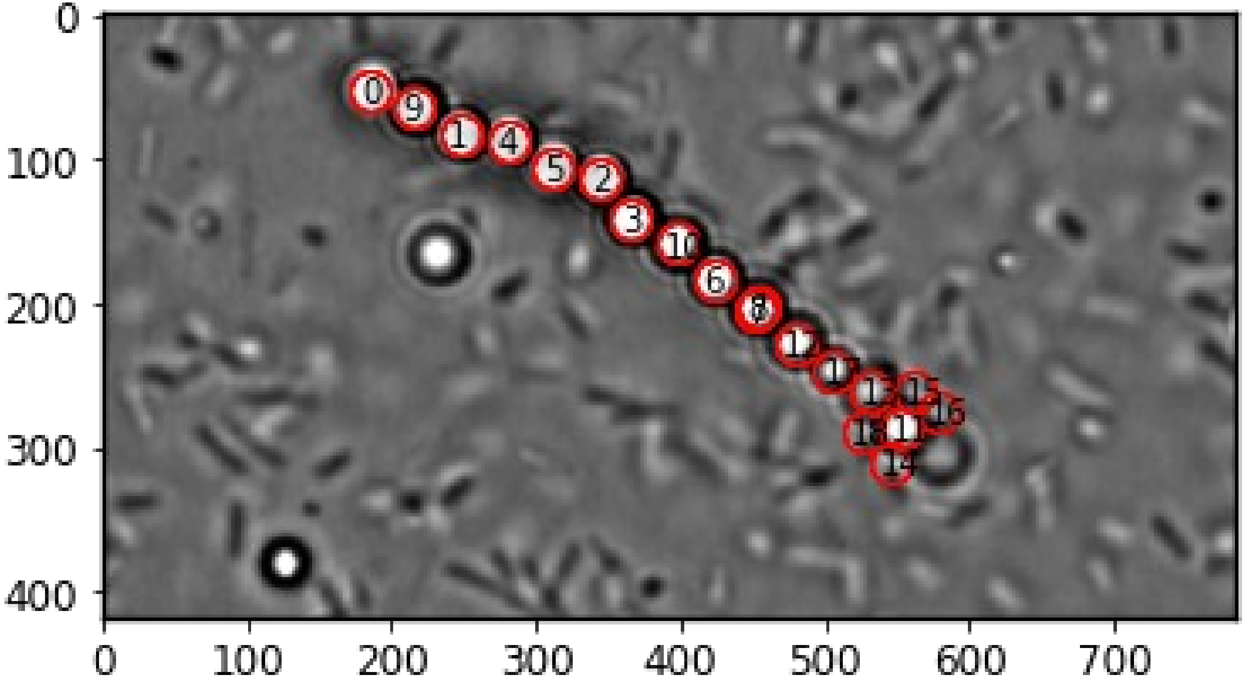
Sorting

- Default: left-right, top-bottom
- Based on total pixel intensity
- Based on cross-correlation with a mask



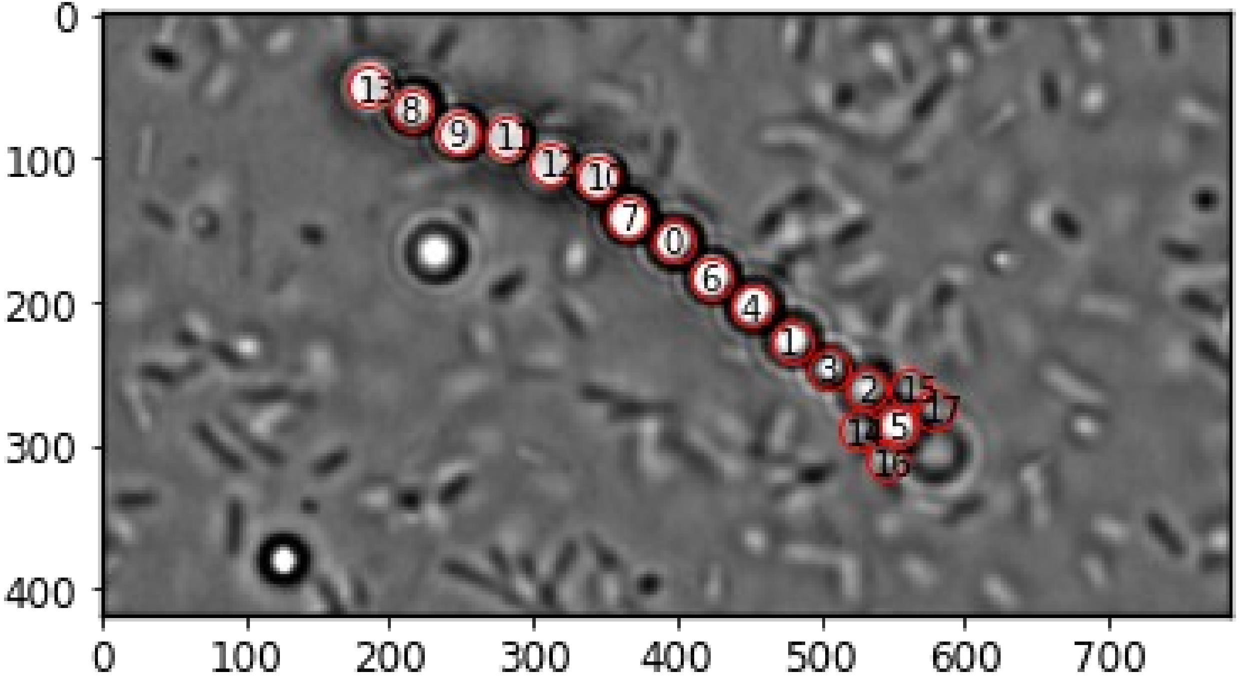
Sorting

- Default: left-right, top-bottom
- Based on total pixel intensity
- Based on cross-correlation with a mask



Sorting

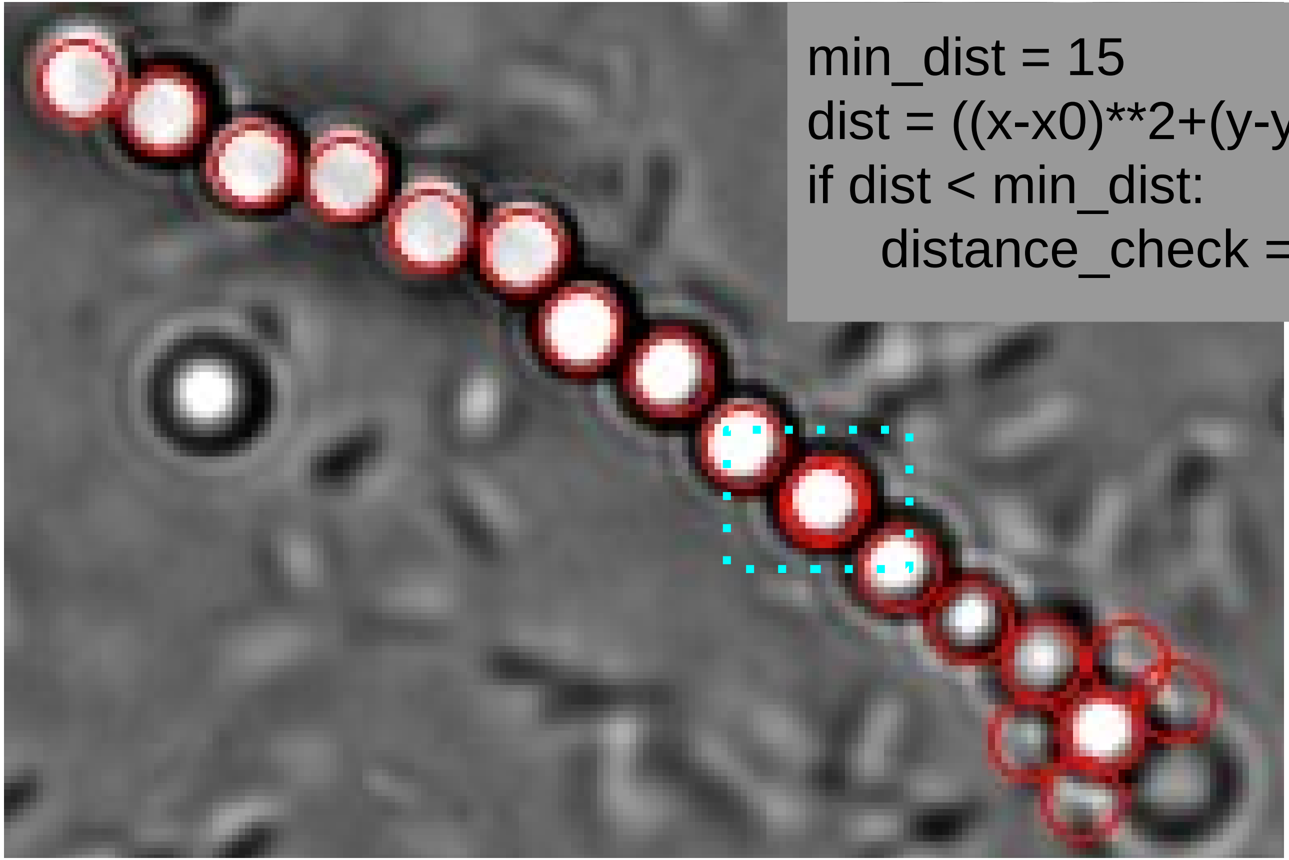
- Default: left-right, top-bottom
- Based on total pixel intensity
- Based on cross-correlation with a mask



Colloidal chain tracking framework

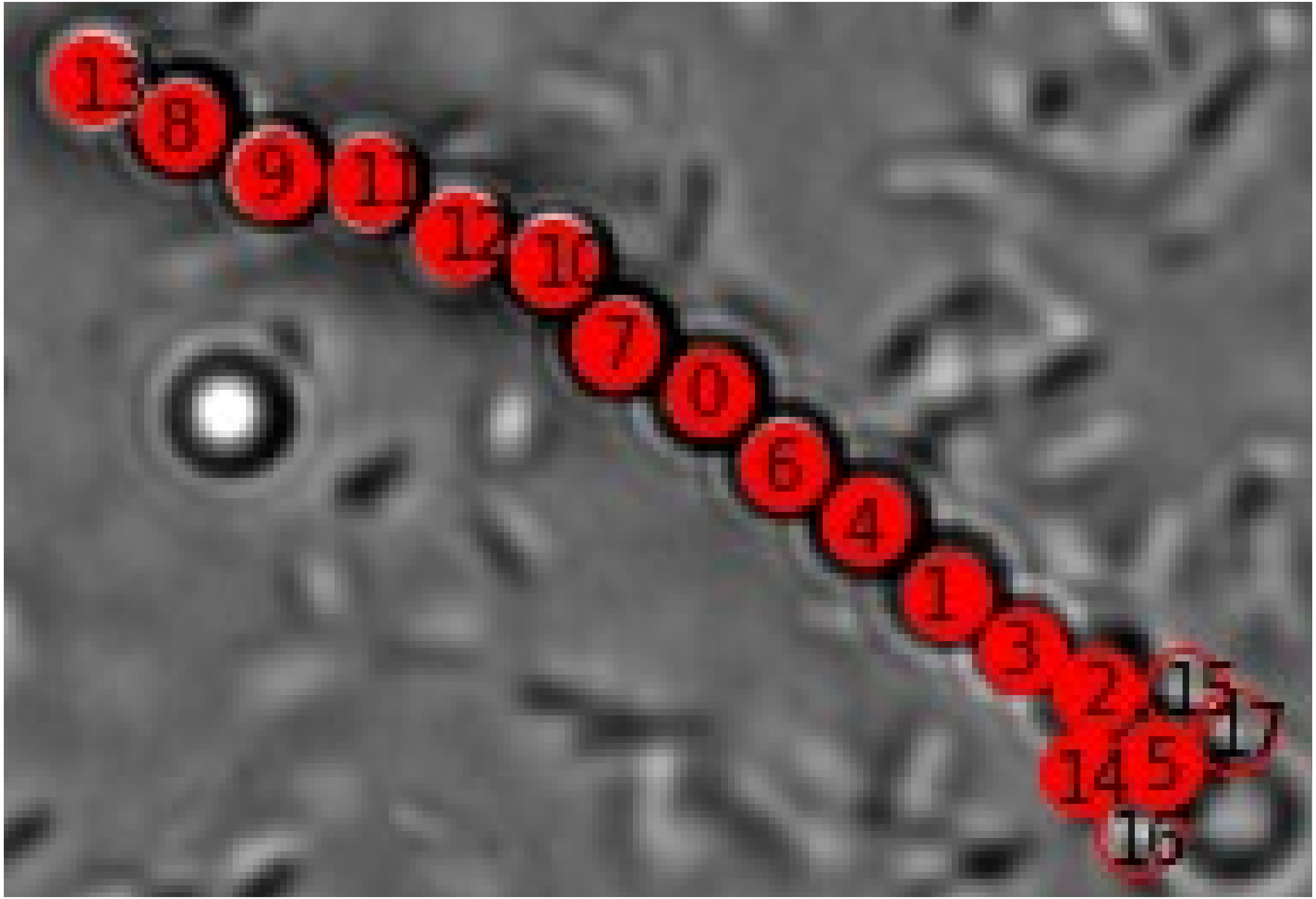
- Preprocessing
- Prelim tracking on dt
- Sorting
  - Total pixel intensity
  - Compare to a mask
- Refine result
  - Distance check
  - Total number of particles
  - Gaussian fitting to get subpixel resolution

Refine - distance check

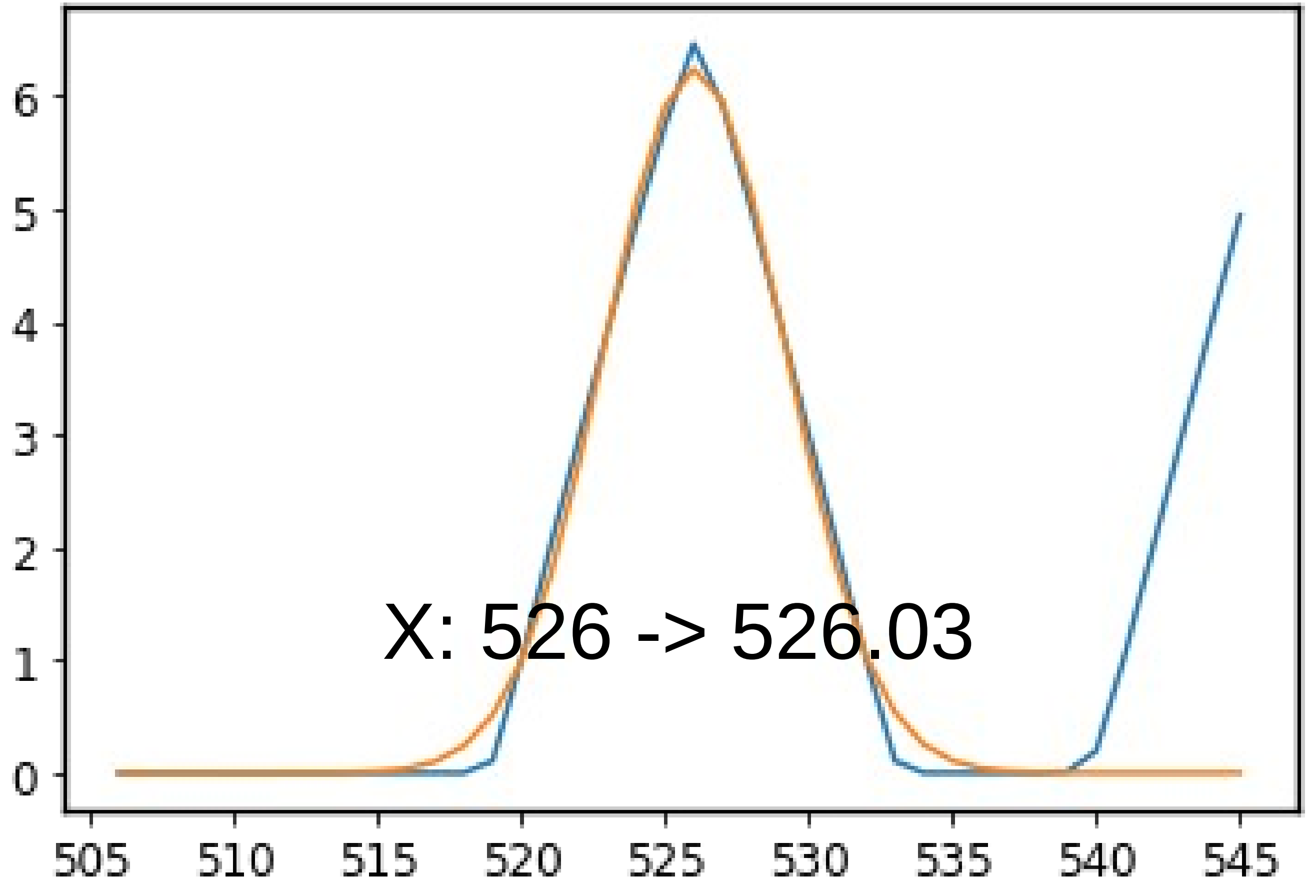


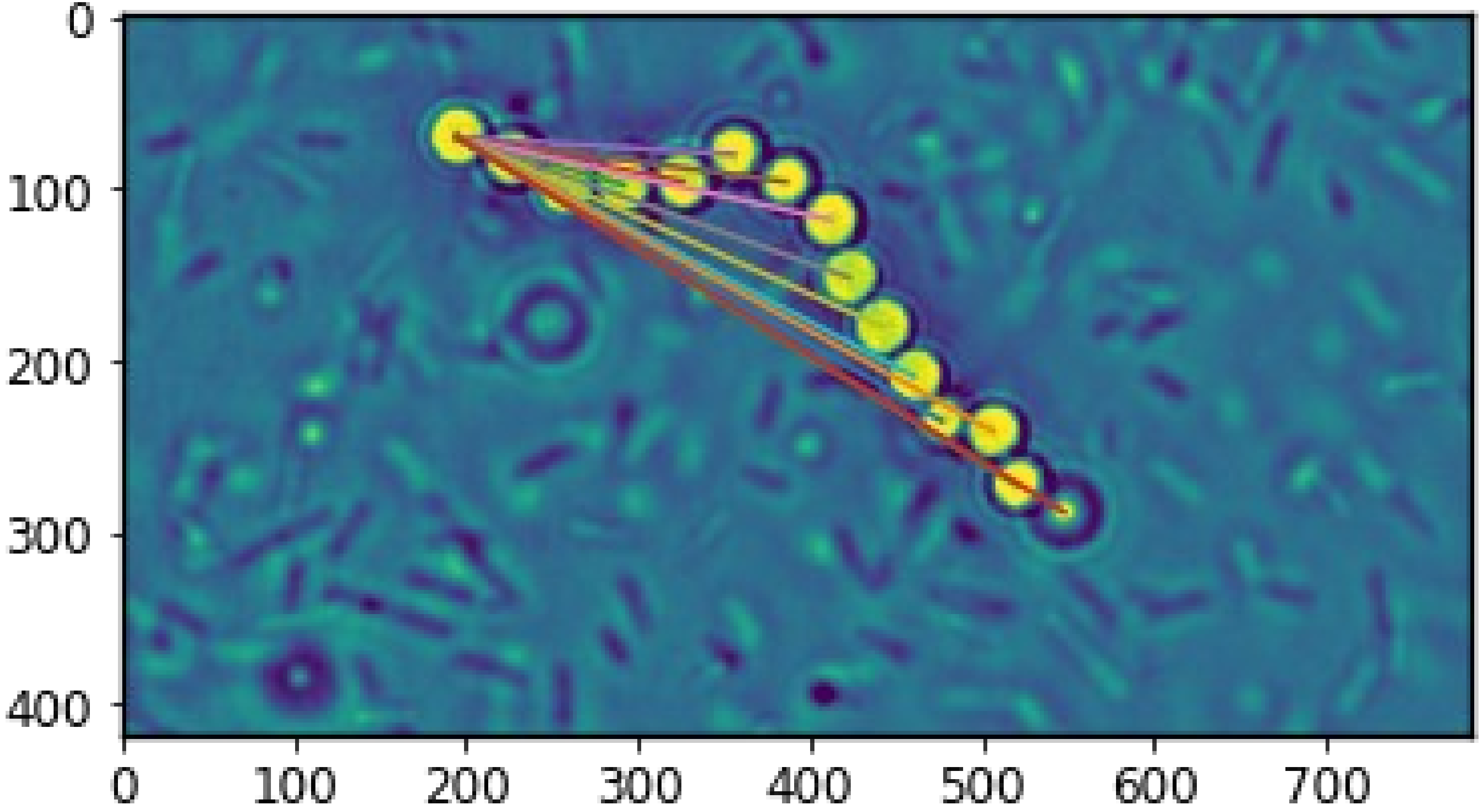
```
min_dist = 15
dist = ((x-x0)**2+(y-y0)**2)**.5
if dist < min_dist:
    distance_check = False
```

Refine - total number of particles = 15



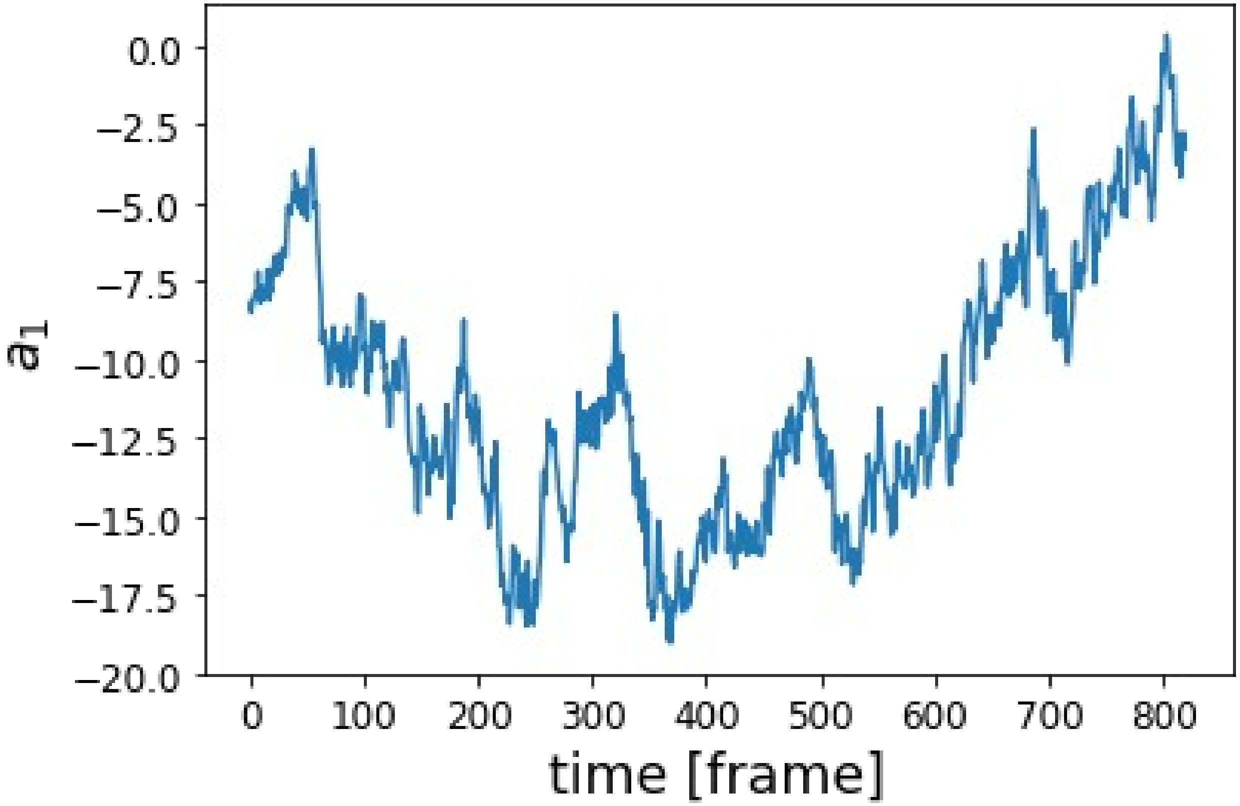
Refine - Gaussian fitting to get subpixel resolution



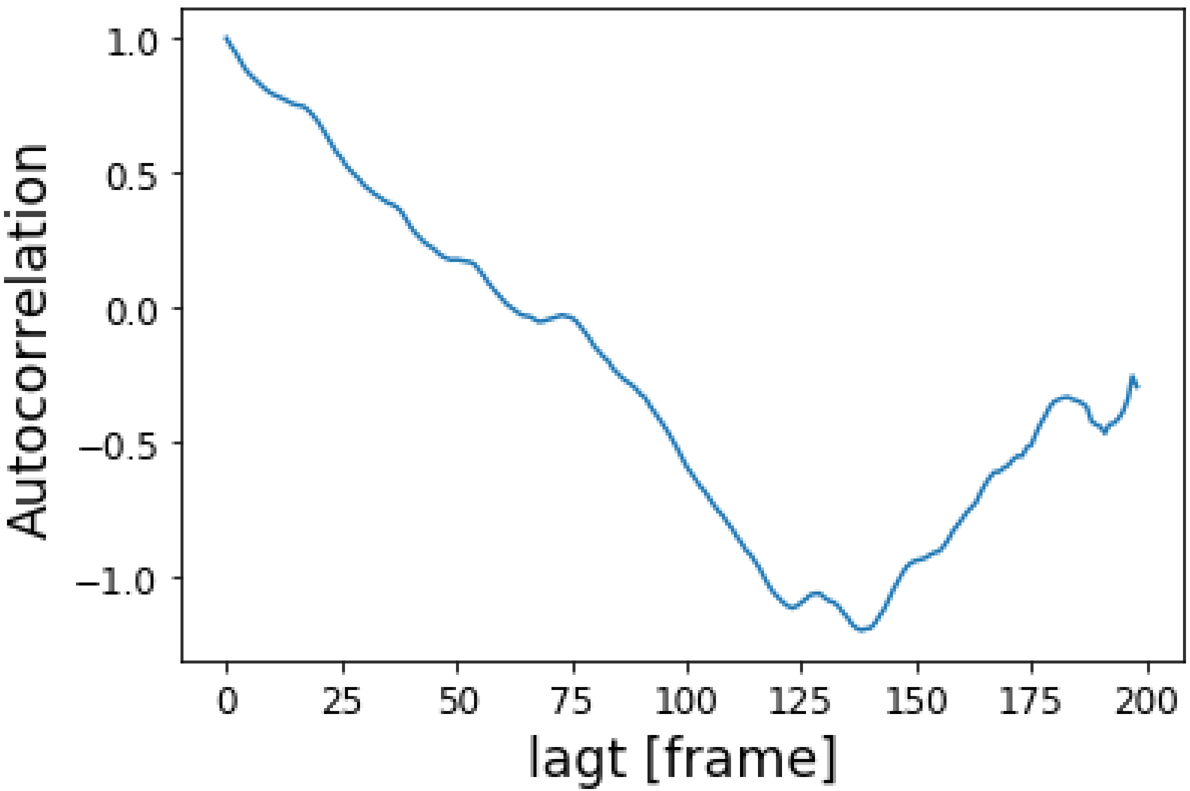

0
100
200
300
400
0
100
200
300
400
500
600
700

$$a_n = \frac{2}{L}\int_0^L \theta(s)\cos\frac{n\pi s}{L}ds$$

$$a_n = \frac{2}{L}\sum_{i=1}^{M}\theta_i \cos\frac{n\pi s_i}{L}\Delta s$$



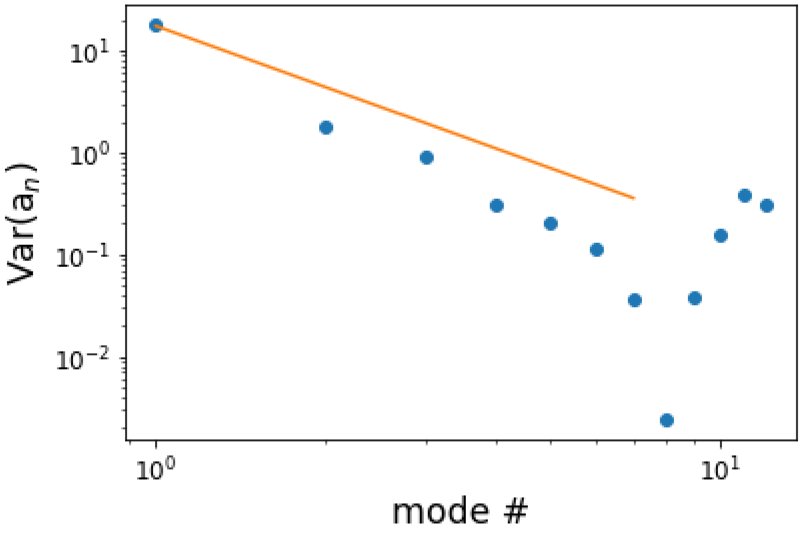
Example of Fourier coefficient time evolution



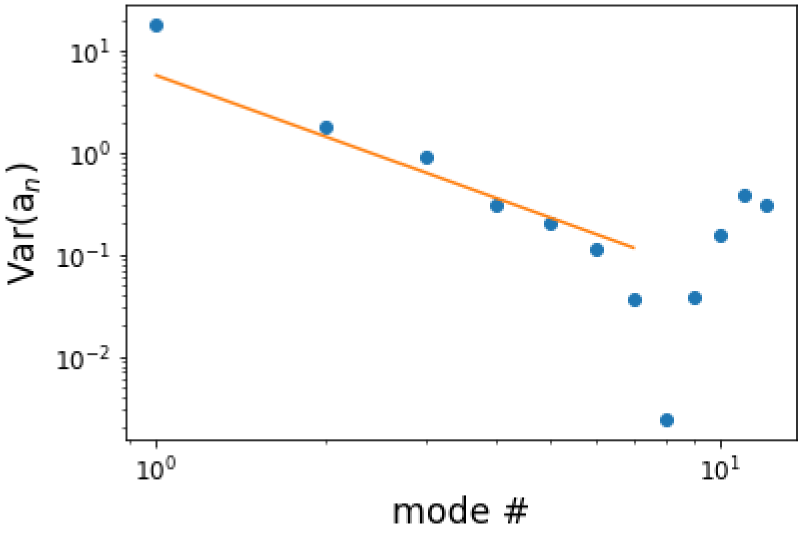
Choose \Delta t to be 100 frames

Persistence length calculation (fix the power of n at -2)

Direct power law fitting



Convert to linear and fit



Converting to linear shows better fitting result

$$l_p = \frac{L^2}{\pi^2 \exp \text{intersect}}$$

$$l_p = 948 \text{ px}$$

94.8 um