DeepTrack - Example 3 - Tracking multiple particles

March 8, 2019

Example 3: Tracking of multiple particles with DeepTrack 1.0

Example code to use DeepTrack to track multiple particles. See also Figure 3.

DeepTrack 1.0
Digital Video Microscopy enhanced with Deep Learning version 1.0
30 November 2018 © Saga Helgadottir, Aykut Argun & Giovanni Volpe
Soft Matter Lab

1. INITIALIZATION

```
In [1]: import deeptrack
```

2. PLAY VIDEO TO BE TRACKED

The video to be tracked is played.

Video file: DeepTrack - Example 3 - Brownian Particles.mp4

Note that the video file must be in the same folder as this notebook.

```
In [2]: %%HTML

        <video width="400" height="400" controls>
        <source src="DeepTrack - Example 3 - Brownian Particles.mp4" type="video/mp4">
        </video>
```

```
<IPython.core.display.HTML object>
```

3. CHECK IMAGE GENERATION ROUTINE

Here, we simulate images of multiple particles similar to those we want to track. In this case, we simulate images with two particles: The first particle position is chosen randomly from a normal distribution with mean of 0 and standard deviation of 10 pixels, and the other is positioned between 15 and 35 pixels from the center. Both particles have a radius between 4 and 6 pixels, and a point-spread function obtained from the combination of a Bessel functions of first and second order of positive and negative intensity respectively. The image background, SNR and gradient intensity are randomly selected from a wide range of values. This results in particle images with a dark ring around a bright center on a bright or dark background with varying SNR and gradient intensity.

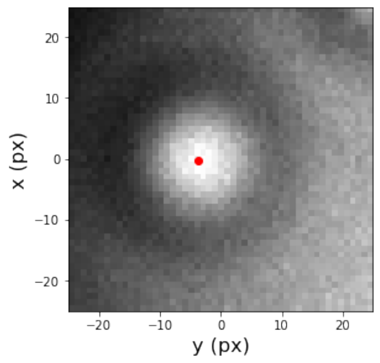
This image generator was used to train the pretraiend network saved in the file "DeepTrack - Example 3 - Pretrained network.h5".

Comments:

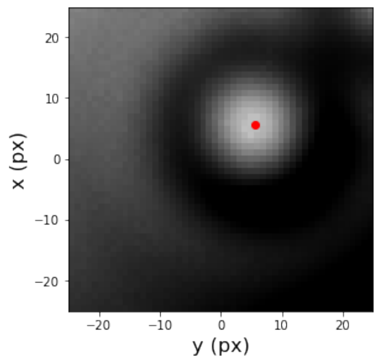
1. The image_parameters_function is a lambda function that determines the kind of particle images for which the deep learning network will be trained. Tuning its parameters is the simplest way to improve the tracking performance.
2. The image_generator is a lambda function that works as image genrator. It does not need to be changed in most cases.
3. The parameter number_of_images_to_show determines the number of sample images that are shown.
4. The red symbol superimposed to the images represents the ground truth particle position.

```
In [3]: ### Define image properties
        %matplotlib inline
        from numpy.random import randint, uniform, normal, choice
        from math import pi

        particle_number = 2

        image_parameters_function = lambda : deeptrack.get_image_parameters(
            particle_center_x_list=lambda : [normal(0 ,10, 1),
                                             choice([int(uniform(-35, -15, 1)), int(uniform(15,
            particle_center_y_list=lambda : [normal(0 ,10, 1),
                                             choice([int(uniform(-35, -15, 1)), int(uniform(15,
            particle_radius_list=lambda : uniform(4, 6, particle_number),
            particle_bessel_orders_list=lambda : [[1, 2], [1, 2]],
            particle_intensities_list=lambda : [[uniform(0.3, 0.7, 1), -uniform(0.2, 0.4, 1)],
                                                [uniform(0.3, 0.7, 1), -uniform(0.3, 0.4, 1)]],
            image_half_size=lambda : 25,
            image_background_level=lambda : uniform(.2, .5),
            signal_to_noise_ratio=lambda : uniform(5, 100),
            gradient_intensity=lambda : uniform(0, 0.8),
            gradient_direction=lambda : uniform(-pi, pi),
            ellipsoidal_orientation=lambda : uniform(-pi, pi, particle_number),
            ellipticity=lambda : 1)

        ### Define image generator
        image_generator = lambda : deeptrack.get_image_generator(image_parameters_function)

        ### Show some examples of generated images
        number_of_images_to_show = 10
```

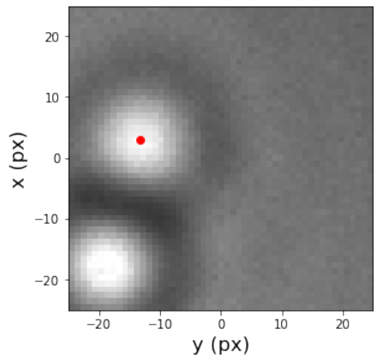
```
for image_number, image, image_parameters in image_generator():
    if image_number>=number_of_images_to_show:
        break

    deeptrack.plot_sample_image(image, image_parameters)
```



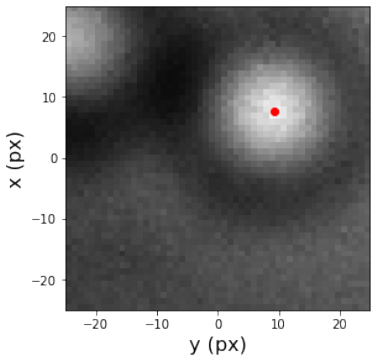
```
particle center x = -0.17 px              image half size = 25.00 px
particle center y = -3.63 px              image background level =  0.43
particle radius =  4.62 px                signal to noise ratio = 26.25
Bessel order =  1.00                      gradient intensity =  0.78
particle intensity =  0.62                gradient direction = -0.34
ellipsoidal_orientation =  2.71
ellipticity =  1.00
```



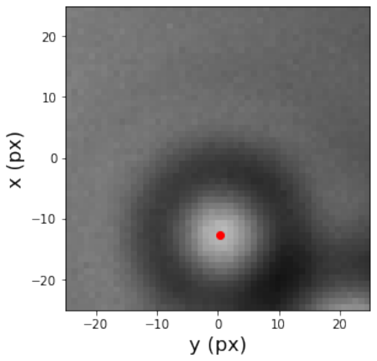
```
particle center x =  5.64 px              image half size = 25.00 px
particle center y =  5.55 px              image background level =  0.22
particle radius =  4.64 px                signal to noise ratio = 80.37
Bessel order =  1.00                      gradient intensity =  0.61
particle intensity =  0.53                gradient direction =  2.19
ellipsoidal_orientation =  0.67
ellipticity =  1.00
```



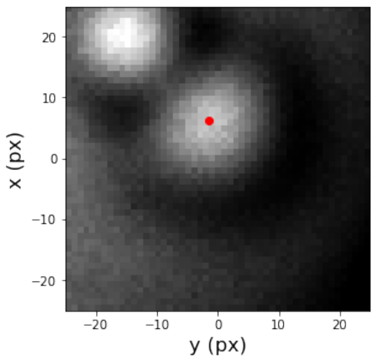
```
particle center x =  2.97 px              image half size = 25.00 px
particle center y = -13.32 px             image background level =  0.47
particle radius =  4.24 px                signal to noise ratio = 62.24
Bessel order =  1.00                      gradient intensity =  0.15
particle intensity =  0.48                gradient direction = -2.54
ellipsoidal_orientation = -2.26
ellipticity =  1.00
```



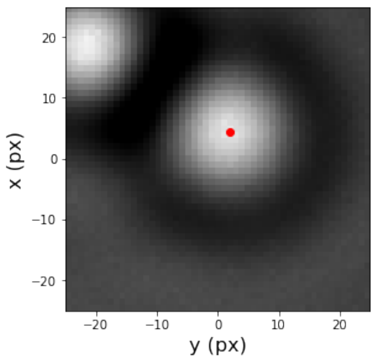
```
particle center x =  7.67 px            image half size = 25.00 px
particle center y =  9.19 px            image background level =  0.34
particle radius =  5.10 px              signal to noise ratio = 32.65
Bessel order =  1.00                    gradient intensity =  0.20
particle intensity =  0.55              gradient direction =  0.51
ellipsoidal_orientation =  0.95
ellipticity =  1.00
```



```
particle center x = -12.71 px           image half size = 25.00 px
particle center y =  0.38 px            image background level =  0.43
particle radius =  4.12 px              signal to noise ratio = 70.01
Bessel order =  1.00                    gradient intensity =  0.18
particle intensity =  0.30              gradient direction =  3.07
ellipsoidal_orientation = -2.93
ellipticity =  1.00
```

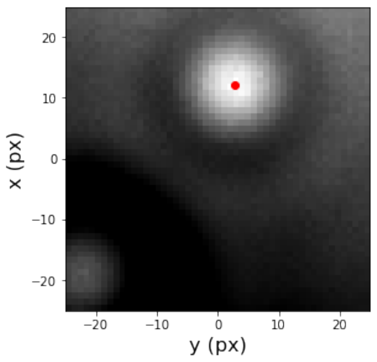


```
particle center x =  6.23 px            image half size = 25.00 px
particle center y = -1.53 px            image background level =  0.25
particle radius =  5.33 px              signal to noise ratio = 26.24
Bessel order =  1.00                    gradient intensity =  0.60
particle intensity =  0.51              gradient direction =  2.76
ellipsoidal_orientation = -0.16
ellipticity =  1.00
```



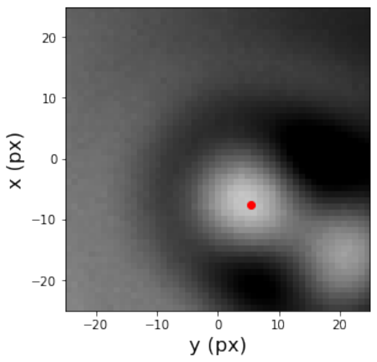
particle center x = 4.37 px
particle center y = 1.97 px
particle radius = 5.44 px
Bessel order = 1.00
particle intensity = 0.62
ellipsoidal_orientation = -2.60
ellipticity = 1.00

image half size = 25.00 px
image background level = 0.26
signal to noise ratio = 81.64
gradient intensity = 0.06
gradient direction = -2.17



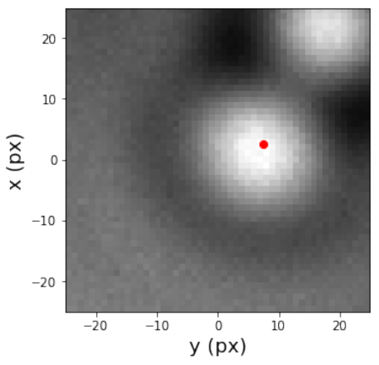
particle center x = 12.17 px
particle center y = 2.70 px
particle radius = 4.01 px
Bessel order = 1.00
particle intensity = 0.69
ellipsoidal_orientation = 3.04
ellipticity = 1.00

image half size = 25.00 px
image background level = 0.20
signal to noise ratio = 49.13
gradient intensity = 0.62
gradient direction = 1.23



particle center x = -7.49 px
particle center y = 5.47 px
particle radius = 4.83 px
Bessel order = 1.00
particle intensity = 0.66
ellipsoidal_orientation = 3.10
ellipticity = 1.00

image half size = 25.00 px
image background level = 0.34
signal to noise ratio = 90.54
gradient intensity = 0.42
gradient direction = -2.59



```
particle center x =  2.50 px          image half size = 25.00 px
particle center y =  7.36 px          image background level =  0.38
particle radius =  5.49 px            signal to noise ratio = 52.34
Bessel order =  1.00                  gradient intensity =  0.26
particle intensity =  0.69            gradient direction = -2.08
ellipsoidal_orientation =  2.59
ellipticity =  1.00
```

4. USE A PRETRAINED DEEP LEARNING NETWORK

The pretraiend network saved in the file "DeepTrack - Example 3 - Pretrained network.h5" is loaded and its performance is tested on a selected video.

Video file: DeepTrack - Example 3 - Brownian Particles.mp4

Note that the file "DeepTrack - Example 3 - Pretrained network.h5" and the video file must be in the same folder as this notebook.

Comments:

1. number_frames_to_be_tracked can be changed to track different number of frames. If number_frames is equal to 0 then the whole video is tracked.
2. box_half_size is half the size of the box to be scanned over the frames. The resulting sample should be comparable to the training image.
3. box_scanning_step is the step that is used to scan the box over the frame. It can be increased for higher accuracy or decreased for lower computational time.

```
In [4]: ### Define the video file to be tracked
        video_file_name = 'DeepTrack - Example 3 - Brownian Particles.mp4'

        ### Define the number of frames to be tracked
        number_frames_to_be_tracked = 2

        ### Define the size of the box to be scanned over the frames
        box_half_size = 25

        ### Define the scanning step over the frame
        box_scanning_step = 5

        ### Preprocess the images
        frame_normalize = 0
        frame_enhance = 1

        ### Load the pretrained network
        saved_network_file_name = 'DeepTrack - Example 3 - Pretrained network.h5'
        network = deeptrack.load(saved_network_file_name)

        ### Track the video
        (number_tracked_frames, frames, predicted_positions_wrt_frame, predicted_positions_wrt_b
            video_file_name,
            network,
            number_frames_to_be_tracked,
            box_half_size,
            box_scanning_step,
            frame_normalize,
            frame_enhance)
```

```
Using TensorFlow backend.
```

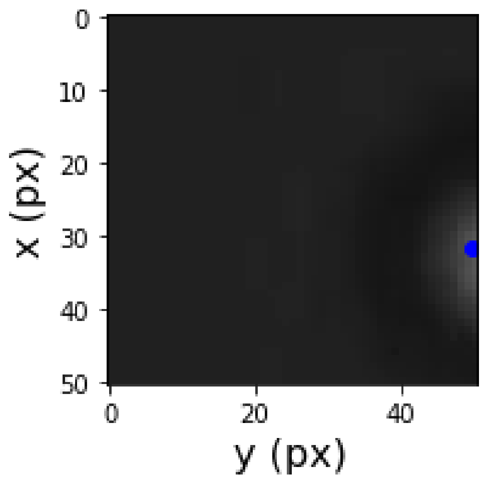
5. SHOW EXAMPLES OF TRACKED SCANNING BOXES

The tracked scanning boxes are plotted over a range of frames, rows and columns.

Comments:

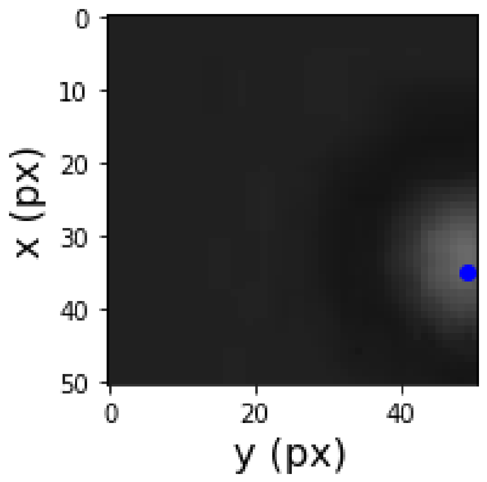
1. frame_to_be_shown can be changed to view different frames.
2. rows_to_be_shown can be changed to view different rows of each of the frames.
3. columns_to_be_shown can be changed to view different columns of each of the frames.
4. The blue symbol is the deep learning network prediction for the position (x, y).

```
In [5]: %matplotlib inline
        ### Define frames, rows and columns of the samples to be shown
        frames_to_be_shown = range(1)
        rows_to_be_shown = range(10,11)
        columns_to_be_shown = range(10)

        ### Show boxes
        deeptrack.plot_tracked_scanning_boxes(
            frames_to_be_shown,
            rows_to_be_shown,
            columns_to_be_shown,
            boxes_all,
            predicted_positions_wrt_box)
```



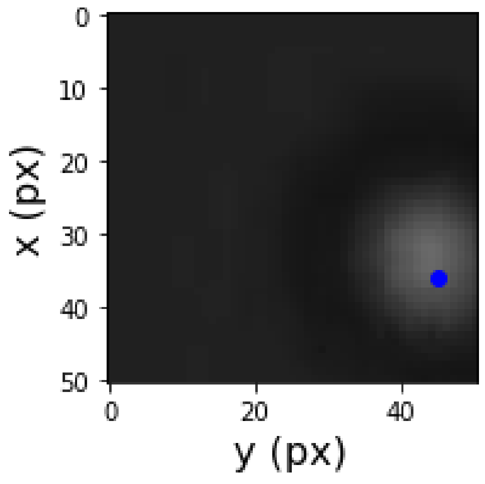
frame = 0
row = 10
column = 0

particle center x = 31.66 px
particle center y = 49.77 px
particle radius = 32.59 px



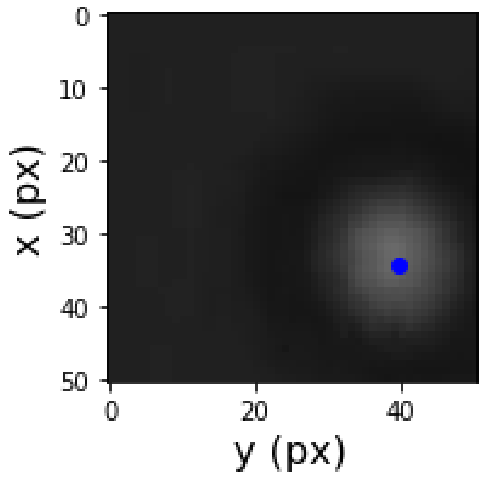
frame = 0
row = 10
column = 1

particle center x = 34.99 px
particle center y = 49.28 px
particle radius = 26.77 px



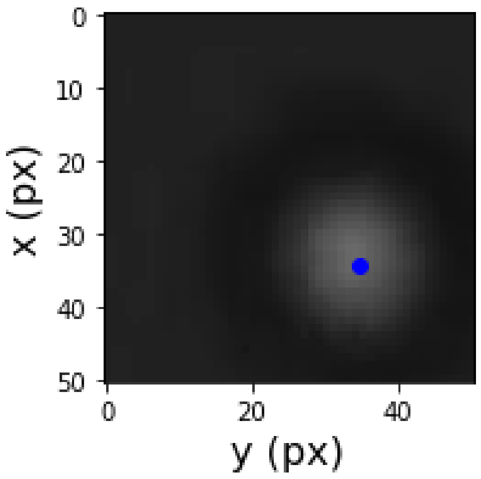
frame = 0
row = 10
column = 2

particle center x = 35.87 px
particle center y = 45.14 px
particle radius = 23.88 px



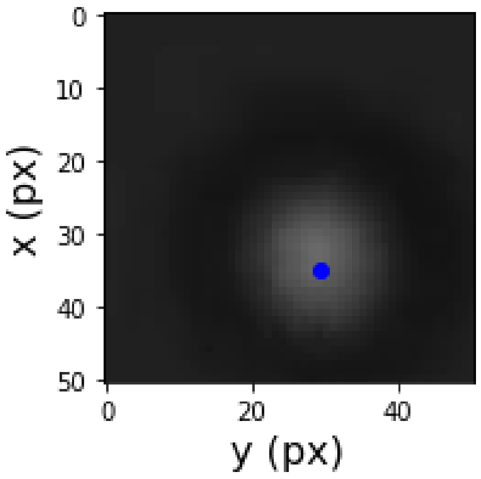
frame = 0
row = 10
column = 3

particle center x = 34.35 px
particle center y = 39.62 px
particle radius = 17.40 px



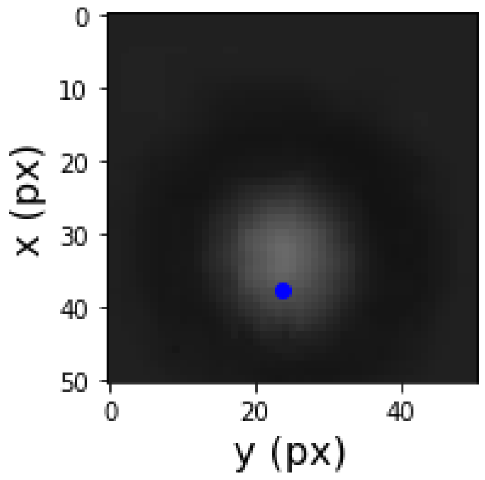
frame = 0
row = 10
column = 4

particle center x = 34.43 px
particle center y = 34.85 px
particle radius = 13.54 px



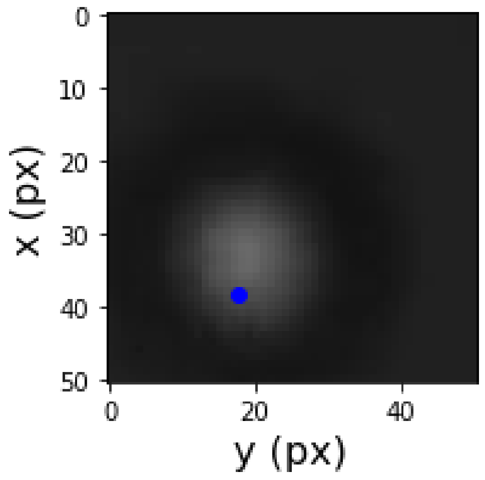
frame = 0
row = 10
column = 5

particle center x = 34.86 px
particle center y = 29.42 px
particle radius = 10.45 px



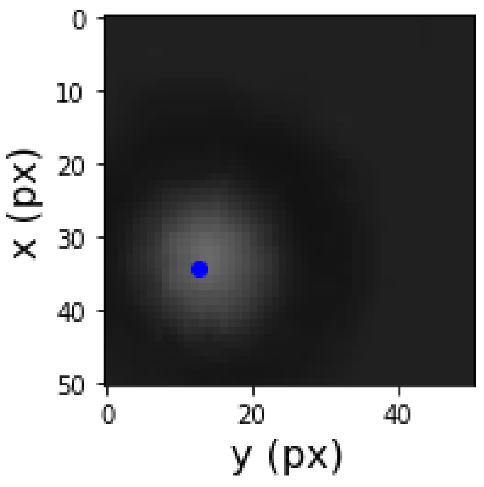
frame = 0
row = 10
column = 6

particle center x = 37.68 px
particle center y = 23.61 px
particle radius = 11.15 px



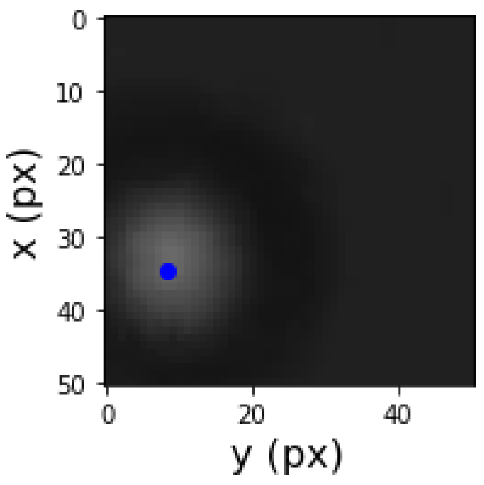
frame = 0
row = 10
column = 7

particle center x = 38.34 px
particle center y = 17.77 px
particle radius = 13.43 px



frame = 0
row = 10
column = 8

particle center x = 34.45 px
particle center y = 12.77 px
particle radius = 15.64 px



frame = 0
row = 10
column = 9

particle center x = 34.70 px
particle center y = 8.22 px
particle radius = 19.48 px

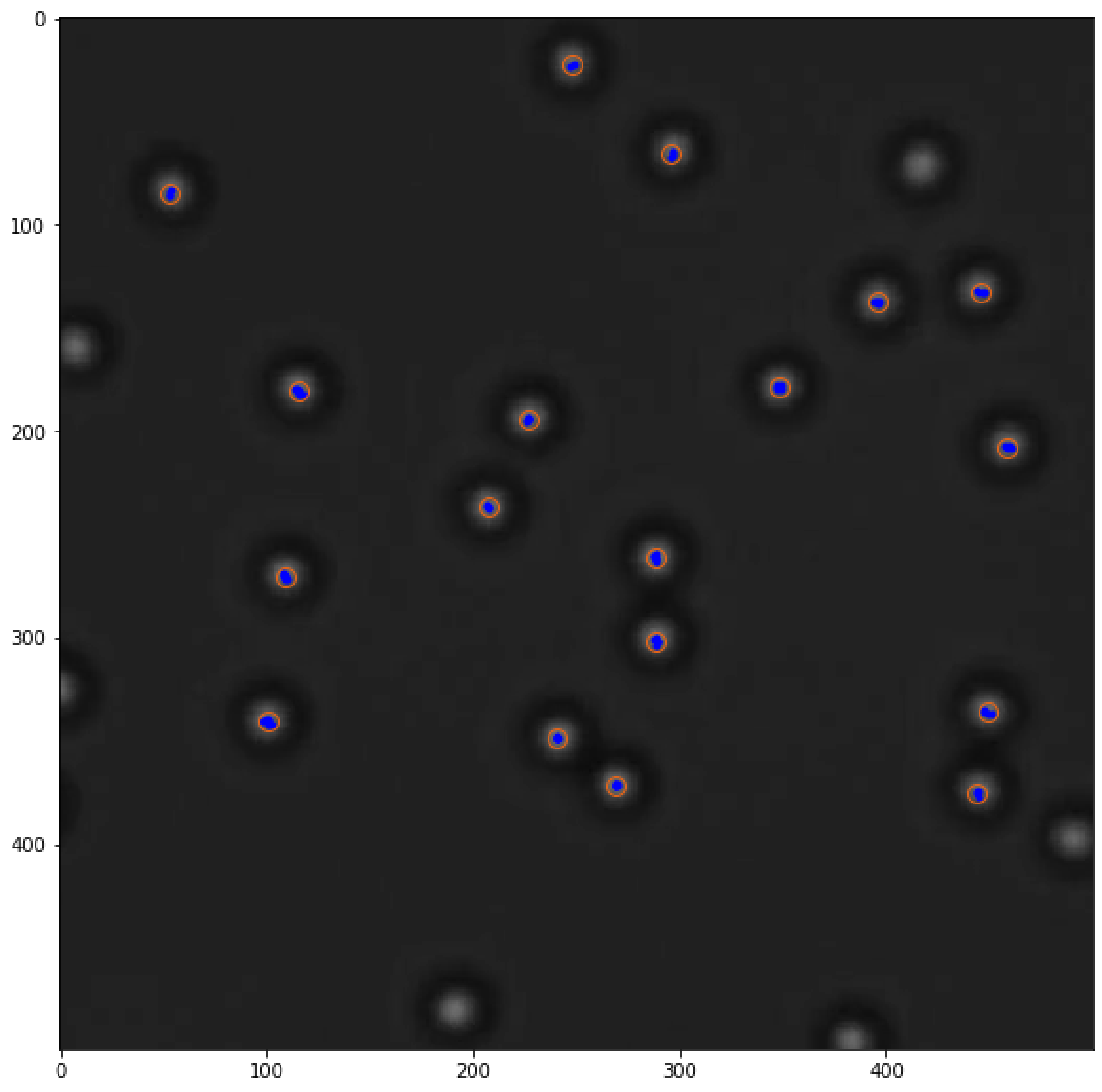
6. SHOW EXAMPLES OF TRACKED FRAMES

The tracked frames are shown.

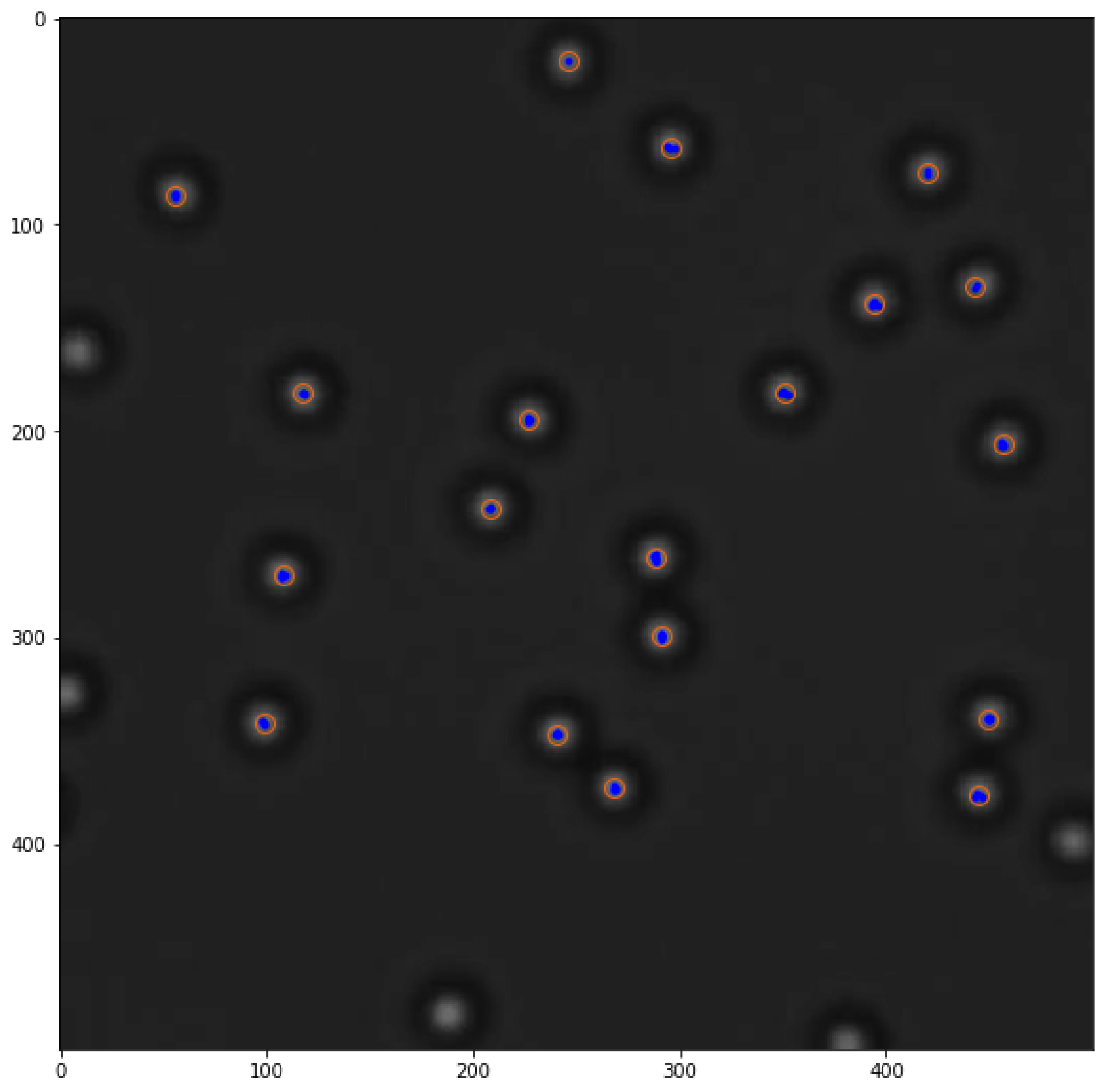
Comments:

1. particle_radial_distance_threshold can be changed to choose which prediction points (blue dots) are to be used to calculate the centroid positions (orange circles). We used 7.5 pixels.
2. particle_maximum_interdistance can be changed to choose what predicted points (blue dots) belong to the same particle. We used 15 pixels.

```
In [6]: ### Define minimum radial distance from the center of the scanning boxes (maximum 10 fra
        particle_radial_distance_threshold = 7.5

        ### Define the minumum distance between predicted scanning points for them belonging to

        particle_maximum_interdistance = 15

        ### Visualize tracked frames
        deeptrack.show_tracked_frames(
            particle_radial_distance_threshold,
            particle_maximum_interdistance,
            number_tracked_frames,
            frames,
            predicted_positions_wrt_frame)
```



0
100
200
300
400
0
100
200
300
400



```
In [ ]:
```