This Report generated at

```
tstamp = datetime('today', 'InputFormat','yyyy-MM-dd');
display(datetime)
```

```
  datetime

   04-Mar-2025 14:47:45
```

Define the source folder here:

```
main_folder =
"C:\Users\al3xm\Documents\_Local_Data\Test_1um_beads_water\simple_tracks"
```

```
main_folder =
"C:\Users\al3xm\Documents\_Local_Data\Test_1um_beads_water\simple_tracks"
```

If you want to scale the tracks, set the scaling factor here in pixels/distance unit. Otherwise, set scaling to -1 to avoid scaling.

```
scaling = 1; % units per input unit0
```

Now, generate the MSD analyzer structure. Note that this filters out any tracks with a track length of less than 7 by default, though this can be changed.

```
ma1 = TrackMateImport(main_folder, true, scaling);
```

```
Warning: Directory already exists.
found 107 tracks in the file.
found a total of 107 tracks in the directory
Warning: Scaling factor active!
Warning: plotting only tracks longer than threshold length
Computing MSD of 104 tracks...   Done.
```

You may override space and time units below as desired. Note that we currently default to frames in pretty much all cases.

```
SpaceUnits = 'microns';
TimeUnits = 'seconds';
display(SpaceUnits)
```

```
SpaceUnits =
'microns'
```

```
display(TimeUnits)
```

```
TimeUnits =
'seconds'
```

Enter the timestep **in time units. Ensure that you use the inverse of the fps, not exposure.**

```
dt=0.01 % 1/(fps)
```

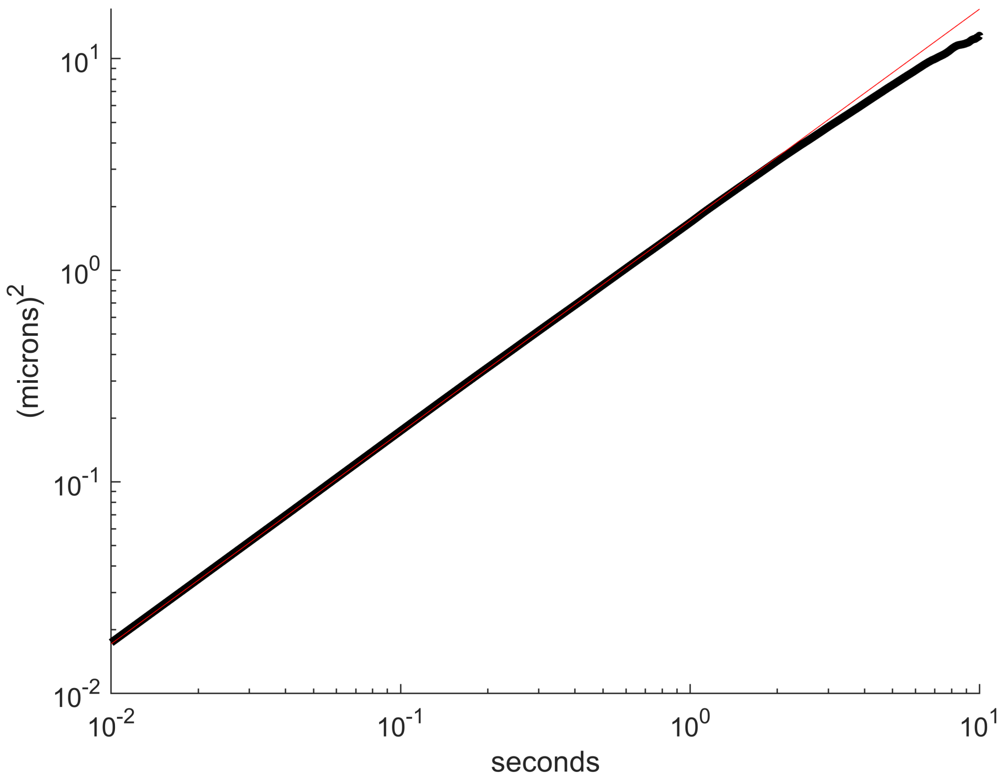
```
dt = 0.0100
```

Now, we plot the MSD. If you want to plot an expected value, set it. Otherwise, set D to -1.

```
D_expect=0.4292;
fprintf(strcat(string(D_expect),' (', SpaceUnits, ')^2/', TimeUnits))
```

```
0.4292 (microns)^2/seconds
```

```
FigMeanMSD(SpaceUnits, TimeUnits,ma1, dt,D_expect, false)
```

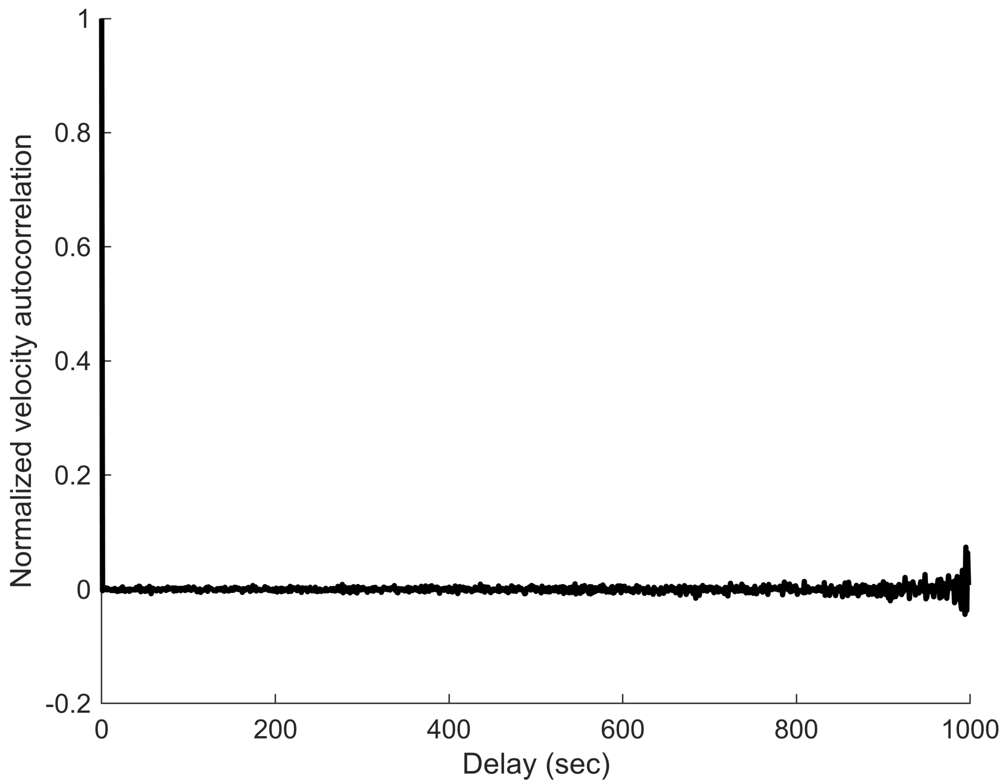



```
ans = 1000×5
         0         0         0   99.5914         0
    0.0100    0.0174    0.0006   99.5635    0.0172
    0.0200    0.0347    0.0015   99.5434    0.0343
    0.0300    0.0521    0.0027   99.5341    0.0515
    0.0400    0.0694    0.0042   99.5247    0.0687
    0.0500    0.0868    0.0059   99.5165    0.0858
    0.0600    0.1043    0.0077   99.5031    0.1030
    0.0700    0.1218    0.0098   99.4929    0.1202
    0.0800    0.1393    0.0119   99.4862    0.1373
    0.0900    0.1568    0.0142   99.4812    0.1545
      ⋮
```

Now we create a plot of the mean velocity correlation as a function of time.

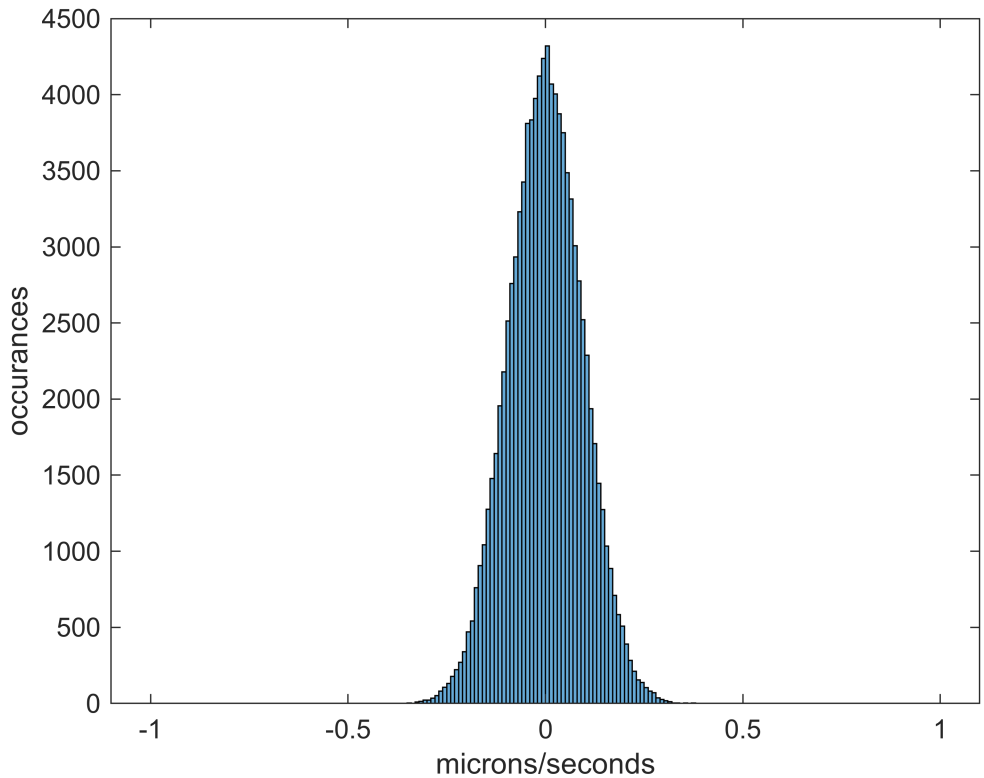
```
figure;
ma1.plotMeanVCorr;
```

```
Computing velocity autocorrelation of 104 tracks... Done.
```



We may also directly compute velocities and plot them as a historgram

```
v = ma1.getVelocities;
  V=vertcat(v{:});
  edges2 = -1:0.01:1;
  histogram(V(:,2),edges2)
  xlabel(strcat(SpaceUnits, '/', TimeUnits))
  ylabel("occurances")
```

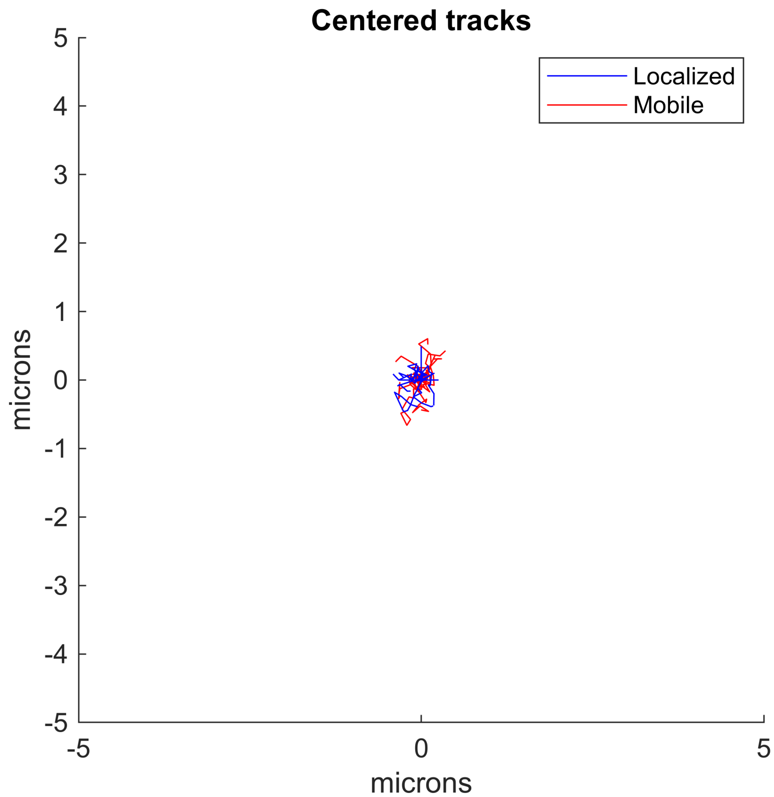


Now, we must call CenterTracks.

```
CenterTracks=CreateCenterTracks(ma1.tracks);
```

We may now create a short vs long track diagram.

```
figure();
UniDomainFigCenterJuxt(SpaceUnits, CenterTracks, 10,10, 10, 10, true, true, 1)
```



Now create a VanHovePlot. This function calls the new VanHove2.m so it is relatively fast and creates bins with equal widths, but it can still take a while to run.

Now, we can can manimputme the CreateVanHovePlots function in a few ways.

We can change the time steps, the number of particles assigned to each bin, and the minimum bin width.

The BinSize determines the number of particles which the Van

```
BinSize = 30
```

```
BinSize = 30
```

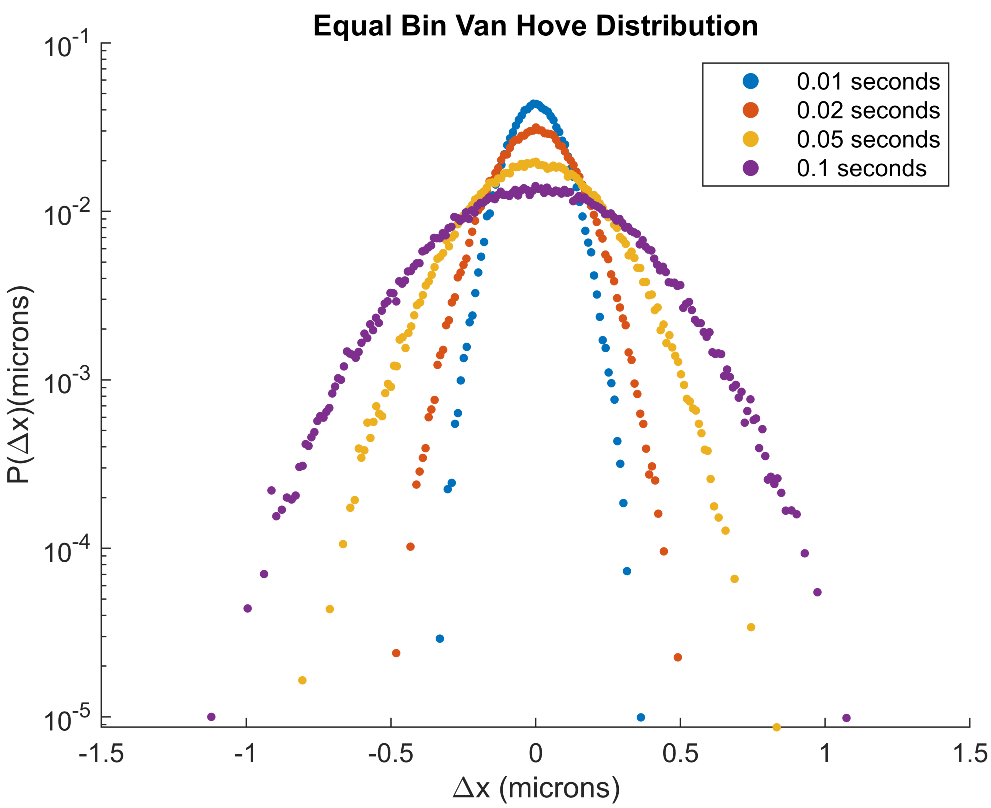
```
MinBin = 0.01 % In microns
```

```
MinBin = 0.0100
```

```
 [CenterPoint,TotStepCount,VanHoveData] = CreateVanHovePlots(SpaceUnits, TimeUnits,
ma1.tracks, BinSize, [1,2,5,10], main_folder, MinBin, dt)
```

```
Calculating VanHove Distribution dt=1

Calculating VanHove Distribution dt=2

Calculating VanHove Distribution dt=5

Calculating VanHove Distribution dt=10
```

```
 Cleaning up for dt=1

 Cleaning up for dt=2

 Cleaning up for dt=5

 Cleaning up for dt=10
Sorting dt=1 into bins
Sorting dt=2 into bins
Sorting dt=5 into bins
Sorting dt=10 into bins
```



CenterPoint = 1×10 cell

...

	1	2	3	4	5	6	7	8	9
1	1×64 double	1×88 double	[]	[]	1×133 double	[]	[]	[]	[]

TotStepCount = 4×2 table

	CellNumber	Value
1	1	97604
2	2	97491
3	5	97174
4	10	96655

VanHoveData = 1×10 cell

...

	1	2	3	4	5	6	7	8
1	97604×4 double	97491×4 double	[]	[]	97174×4 double	[]	[]	[]

Save this file as a pdf.