1um beads imaged at 6fps on OpenFlexure scope

This Report generated at

```
disp(datetime)
```

```
    06-Mar-2025 09:10:08
```

Define the source folder here:

```
main_folder = "C:\Users\al3xm\Downloads\realconverted_tracks\realconverted_tracks"
```

```
main_folder =
"C:\Users\al3xm\Downloads\realconverted_tracks\realconverted_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 unit
```

```
scaling = 1
```

Define the units

```
SpaceUnits = 'microns';
TimeUnits = 'seconds';
```

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

```
dt=0.1667 % 1/(fps)
```

```
dt = 0.1667
```

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 389 tracks in the file.
found 393 tracks in the file.
found 418 tracks in the file.
found a total of 418 tracks in the directory
Warning: Scaling factor active!
Warning: plotting only tracks longer than threshold length
Computing MSD of 890 tracks...   Done.
```

```
ma1 =ma1.fitMSD(0.25); % set to short clipping factor to find diffusive rate
```

```
Fitting 890 curves of MSD = f(t), taking only the first 25% of each curve...   Done.
```

```
ma1 = ma1.fitLogLogMSD(0.75); % set to long clipping factor for linearity measure
```

```
Fitting 890 curves of log(MSD) = f(log(t)), taking only the first 75% of each curve... Done.
```

```
d_est = mean(ma1.lfit.a) * dt; % gives the estimated slope of the MSD in units^2/
second
```

```
gamma_est = mean(ma1.loglogfit.alpha) % Gives time scaling factor of the MSD.
```

```
gamma_est = 0.8582
```

```
fprintf(strcat("The diffusion coefficient is D=", string(d_est), ' ', SpaceUnits,
'^2/', TimeUnits));
```

```
The diffusion coefficient is D=0.043552microns^2/seconds
```

```
fprintf(strcat("The time scaling factor is gamma=", string(gamma_est), '.'));
```

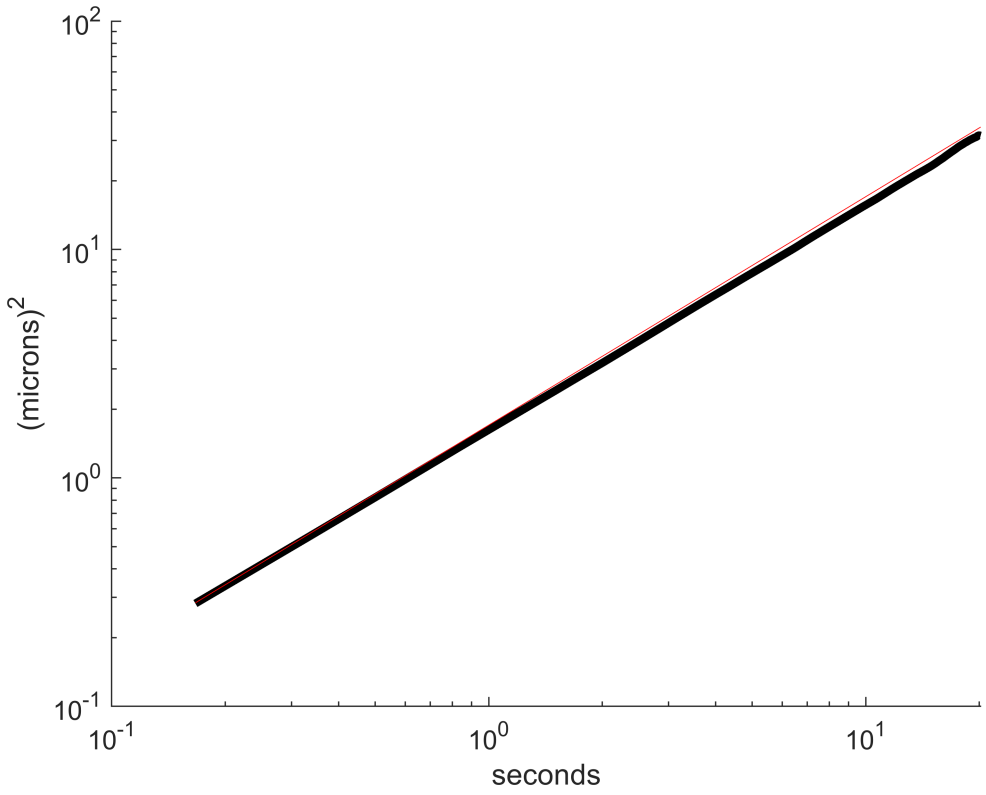
```
The time scaling factor is gamma=0.85821.
```

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

```
D_expect=0.426;
fprintf(strcat('excpected diffusion coefficient:', string(D_expect),' (',
SpaceUnits, ')^2/', TimeUnits))
```

```
excpected diffusion coefficient:0.426 (microns)^2/seconds
```

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



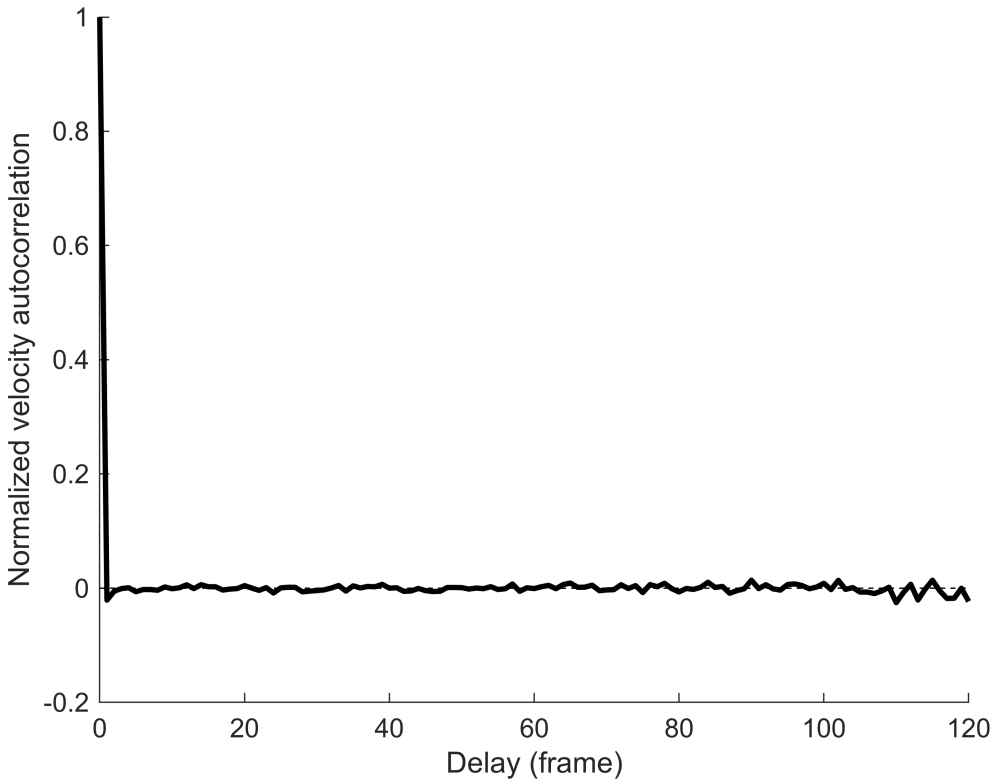

```
ans = 122×5
         0         0         0  648.9517         0
    0.1667    0.2822    0.0695  634.7323    0.2841
    0.3334    0.5534    0.1326  624.6791    0.5681
    0.5001    0.8227    0.2042  620.2755    0.8522
    0.6668    1.0920    0.2820  616.8273    1.1362
    0.8335    1.3609    0.3690  613.9188    1.4203
    1.0002    1.6253    0.4582  611.1059    1.7043
    1.1669    1.8874    0.5504  607.9886    1.9884
```

```
    1.3336    2.1496    0.6527  604.5522    2.2725
    1.5003    2.4091    0.7627  601.8141    2.5565
     ⋮
```

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

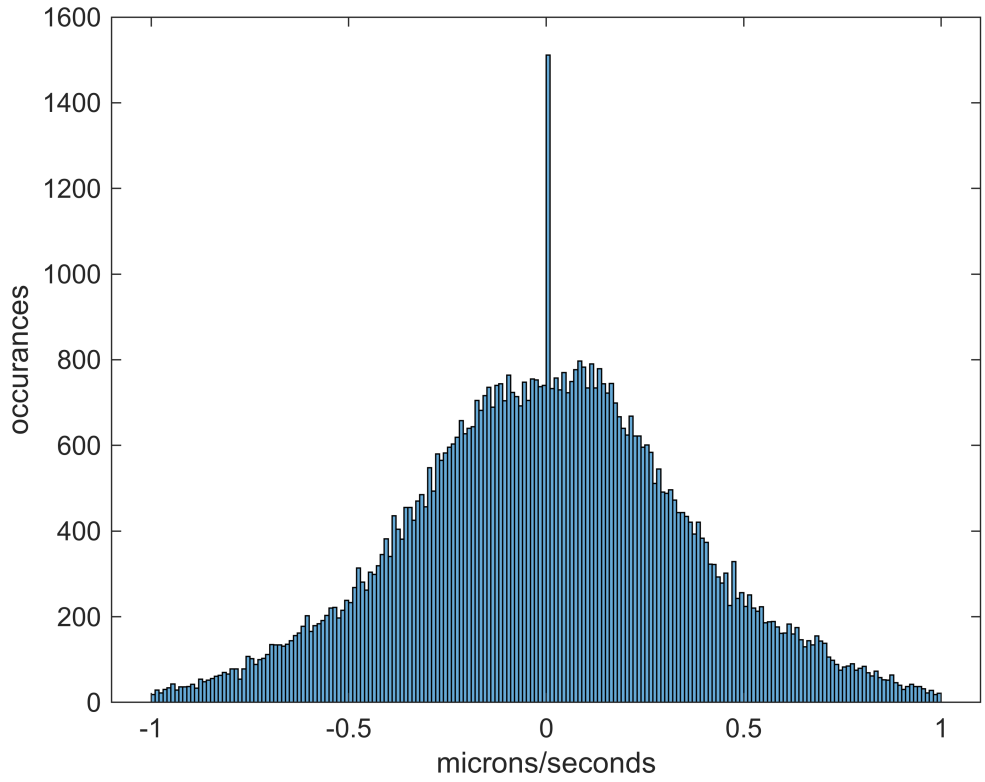
```
figure;
ma1.plotMeanVCorr;
```

```
Computing velocity autocorrelation of 890 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 = 10
```

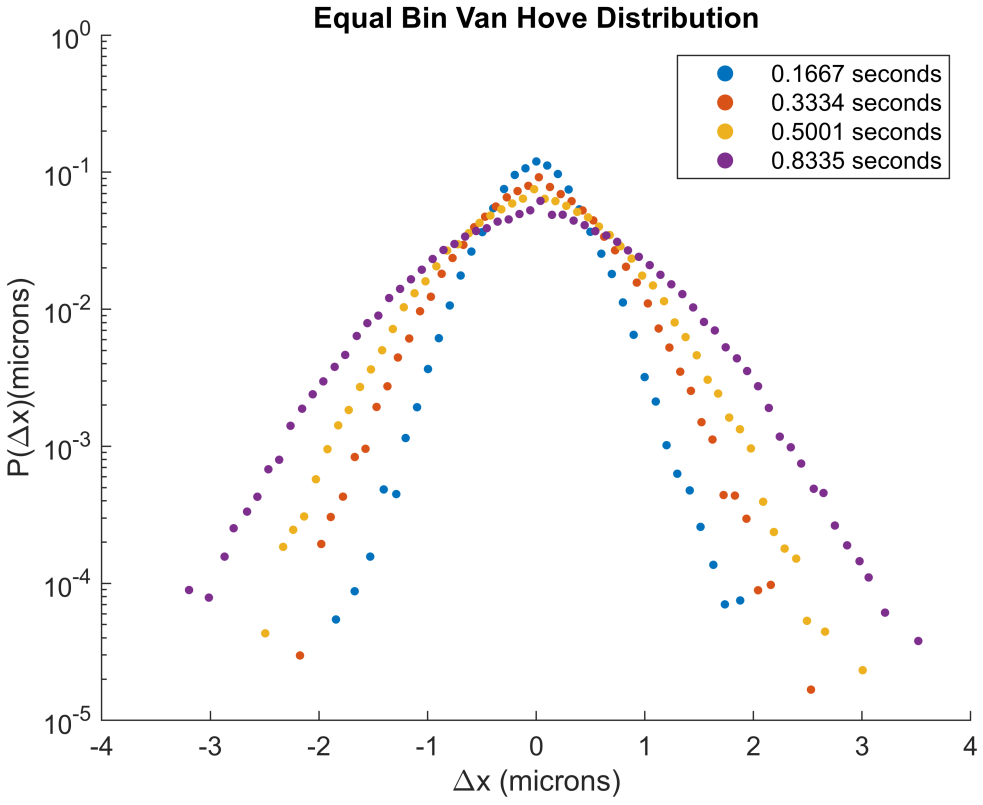
```
BinSize = 10
```

```
MinBin = 0.1 % In microns
```

```
MinBin = 0.1000
```

```
[CenterPoint,TotStepCount,VanHoveData] = CreateVanHovePlots(SpaceUnits, TimeUnits, ma1.tracks, BinSize, [1,2,3,5], main_folder, MinBin, dt)
```

```
Calculating VanHove Distribution dt=1

Calculating VanHove Distribution dt=2

Calculating VanHove Distribution dt=3

Calculating VanHove Distribution dt=5

Cleaning up for dt=1

Cleaning up for dt=2

Cleaning up for dt=3

Cleaning up for dt=5
Sorting dt=1 into bins
Sorting dt=2 into bins
Sorting dt=3 into bins
Sorting dt=5 into bins
```



CenterPoint = 1×5 cell

	1	2	3	4	5
1	1×36 double	1×44 double	1×52 double	[]	1×64 double

TotStepCount = 4×2 table

	CellNumber	Value
1	1	66906
2	2	65605

	CellNumber	Value
3	3	64708
4	5	63037

VanHoveData = 1×5 cell

	1	2	3	4	5
1	66906×4 double	65605×4 double	64708×4 double	[ ]	63037×4 double

Save this file as a pdf.