Qdots in PAAm at 389fps

To run *and* export, paste the following command. Otherwise, export manually. You must first define monkier to do this.

```
% export("TrackMate_Live_Script_guide.mlx",
strcat('C:\Users\al3xm\Documents\GitHub\HISTia\TrackMate Reports\',monkier),
HideCode=false, OpenExportedFile=true, Run=true,FigureResolution=1600)
```

Name this data analysis run

```
monkier = '389fps Qdots In PAAm March 13 Dataset';
disp(monkier)
```

```
389fps Qdots In PAAm March 13 Dataset
```

Set the current directory for data saving purposes.

```
cd('C:\Users\al3xm\Documents\GitHub\HISTia')
```

This Report generated at

```
disp(datetime)
```

```
   24-Mar-2025 11:16:38
```

Define the source folder here:

```
main_folder =
"C:\Users\al3xm\Documents\_Local_Data\25.03.12_PAG_HILO_Beads\2.5msTracks"
```

```
main_folder =
"C:\Users\al3xm\Documents\_Local_Data\25.03.12_PAG_HILO_Beads\2.5msTracks"
```

```
paths =  FileFinder(main_folder, '.xml')
```

```
paths = 1×36 string
"C:\Users\al3xm\Documents\_Local_Data\25.03.12_PAG_HILO_Beads\2.5msTrac…  "C:\Us ⋯
```

```
if ~isfolder(main_folder)
    error('please enter a valid directory')
elseif isempty(paths)
    error('please ensure the folder contains .xml ParticleTrack exports')
end
```

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=0.00257254 % 1/(fps)
```

```
dt = 0.0026
```

what is the minimum track length you want to consider?

```
MinTrackLength = 7
```

```
MinTrackLength = 7
```

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, MinTrackLength, scaling);
```

```
Warning: Directory already exists.
found 1280 tracks in the file.
found 1106 tracks in the file.
found 1037 tracks in the file.
found 985 tracks in the file.
found 1086 tracks in the file.
found 892 tracks in the file.
found 877 tracks in the file.
found 1336 tracks in the file.
found 2179 tracks in the file.
found 1768 tracks in the file.
found 1929 tracks in the file.
found 1752 tracks in the file.
found 1441 tracks in the file.
found 1425 tracks in the file.
found 1479 tracks in the file.
found 1431 tracks in the file.
found 1540 tracks in the file.
found 1288 tracks in the file.
found 1422 tracks in the file.
found 996 tracks in the file.
found 1288 tracks in the file.
found 1180 tracks in the file.
found 1027 tracks in the file.
found 900 tracks in the file.
found 898 tracks in the file.
found 926 tracks in the file.
found 853 tracks in the file.
found 1001 tracks in the file.
found 1270 tracks in the file.
found 1055 tracks in the file.
found 1518 tracks in the file.
found 1425 tracks in the file.
found 1415 tracks in the file.
found 1415 tracks in the file.
found 1842 tracks in the file.
found 2116 tracks in the file.
found a total of 2116 tracks in the directory
Warning: Scaling factor active!
Warning: plotting only tracks longer than threshold length
Computing MSD of 20842 tracks... 20Done.
```

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

```
Fitting 20842 curves of MSD = f(t), taking only the first 25% of each curve... 20Done.
```

```
ma1 = ma1.fitLogLogMSD(0.25); % set to long clipping factor for linearity measure
```

```
Fitting 20842 curves of log(MSD) = f(log(t)), taking only the first 25% of each curve... Done.
```

```
d_est = mean(ma1.lfit.a) * (1/dt); % starts with units^2 per frame, divide by
frames/time unit.
% This gives the estimated slope of the MSD in units^2/second
gamma_est = mean(ma1.loglogfit.alpha); % Gives time scaling factor of the MSD.
dimensionality_modifier = 4; % Conversion to use conventional x^2 = 4Dt metric for
2D diffusion.
fprintf(strcat("The diffusion coefficient is D=", string(d_est/
(dimensionality_modifier)), ' ', SpaceUnits, '^2/', TimeUnits));
```

```
The diffusion coefficient is D=5.6786microns^2/seconds
```

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

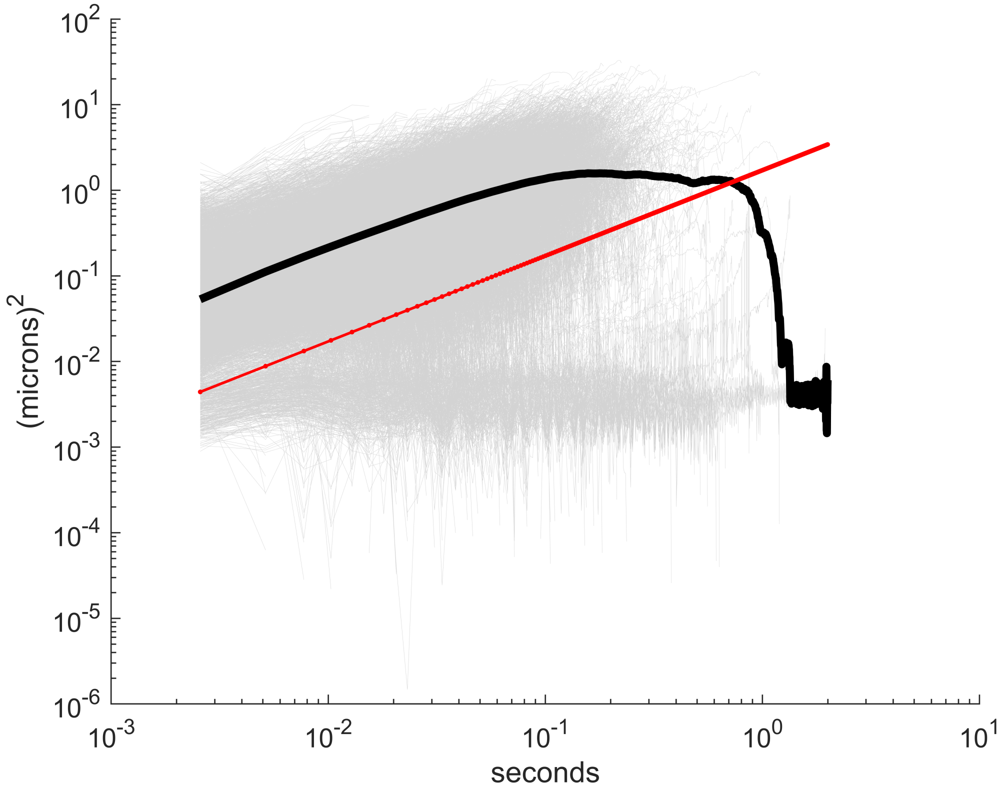
```
The time scaling factor is gamma=0.88981.
```

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

```
D_expect=0.4292; %D=0.4292 for 1um beads in water.
fprintf(strcat('excpected diffusion coefficient: ', string(D_expect),' (',
SpaceUnits, ')^2/', TimeUnits))
```

```
excpected diffusion coefficient:0.4292 (microns)^2/seconds
```

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

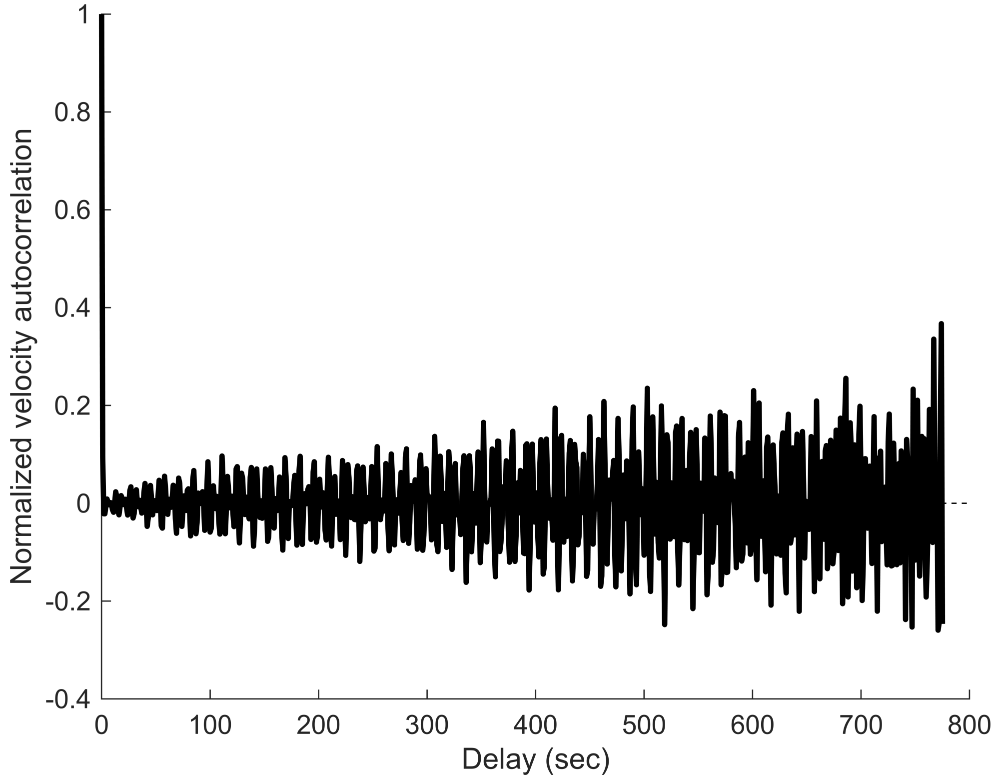



```
ans = 777×5
10^3 ×
        0         0         0    9.8315         0
   0.0000    0.0001    0.0001    8.8477    0.0000
   0.0000    0.0001    0.0001    7.7853    0.0000
   0.0000    0.0002    0.0002    7.1727    0.0000
   0.0000    0.0002    0.0002    6.7142    0.0000
   0.0000    0.0003    0.0002    6.3042    0.0000
   0.0000    0.0003    0.0003    5.9522    0.0000
   0.0000    0.0004    0.0003    5.5838    0.0000
   0.0000    0.0004    0.0004    5.2018    0.0000
   0.0000    0.0005    0.0004    4.8248    0.0000
     ⋮
```

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

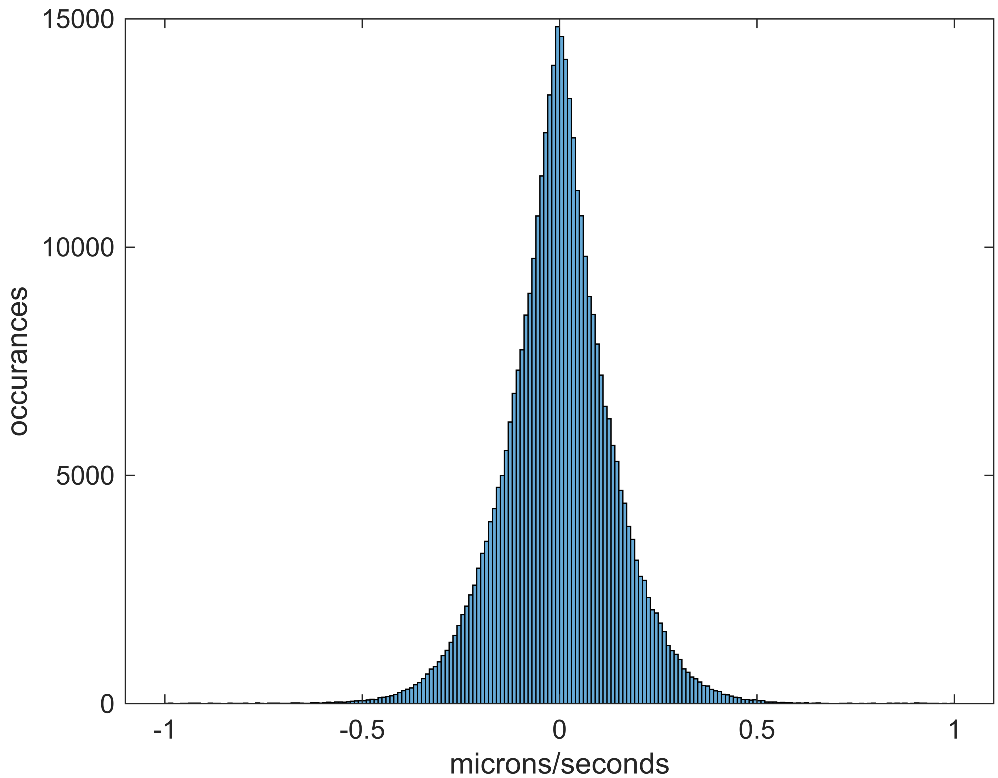
```
figure;
ma1.plotMeanVCorr;
```

```
Computing velocity autocorrelation of 20842 tracks...
```



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
```

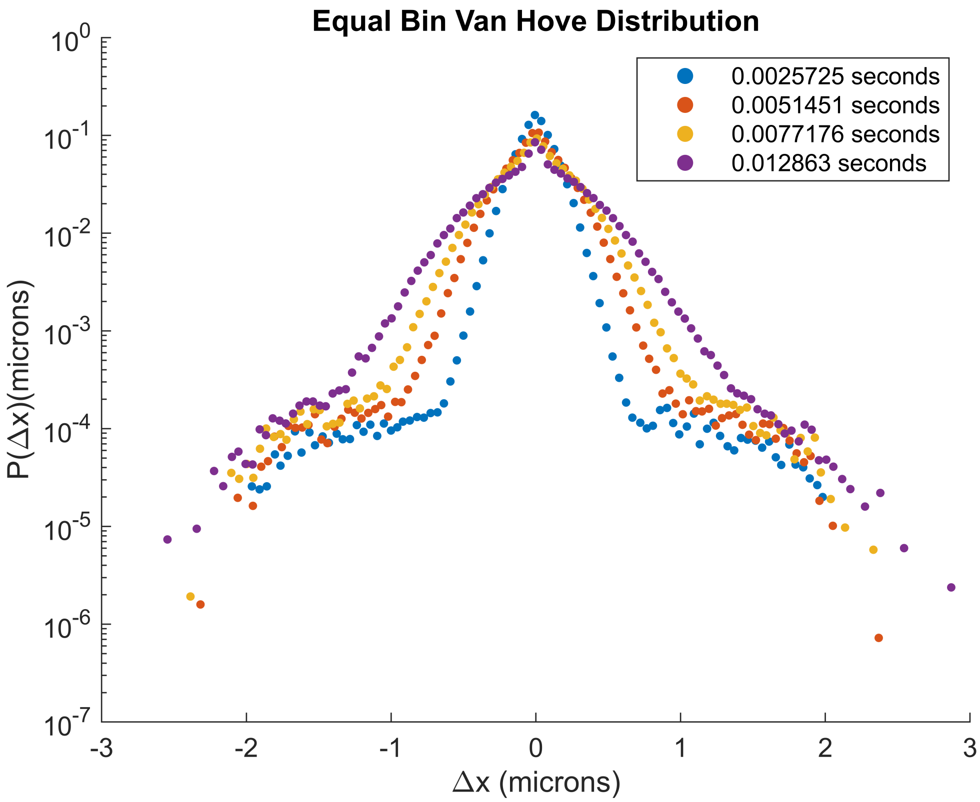
```
BinSize = 10
```

```
MinBin = 0.045 % In microns. I don't recommend going much below diffraction limit/4
```

```
MinBin = 0.0450
```

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

```
 Calculating VanHove Distribution dt=1

5000
10000
15000
20000
 Calculating VanHove Distribution dt=2

5000
10000
15000
20000
 Calculating VanHove Distribution dt=3

5000
10000
15000
20000
 Calculating VanHove Distribution dt=5

5000
10000
15000
20000
 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
```




```
VanHoveStats = struct with fields:
             Data: {5×1 cell}
      CenterPoint: {5×1 cell}
         EquiProb: {5×1 cell}
    TotStepsCount: [4×2 table]
          BinSize: 10
              tau: [1 2 3 5]
        FrameTime: 0.0026
           MinBin: 0.0450
```

Save the data in SuperStruct format to the SuperStructs Github folder.

```
MasterSaveD(monkier, ma1,main_folder, MinTrackLength, SpaceUnits, TimeUnits,
VanHoveStats, cd)
```

```
ans =
"C:\Users\al3xm\Documents\GitHub\HISTia\SuperStructs\389fps Qdots In PAAm March 13 Dataset2025-03-24.11.21.15.mat"
```

Save this output as a pdf for later viewing.