Analysis of 605nm Qdots imaged at 388fps on Hestia

Name this data analysis run

```
monkier = 'Analysis of 605nm Qdots imaged at 388fps on Hestia';
disp(monkier)
```

```
Analysis of 605nm Qdots imaged at 388fps on Hestia
```

This Report generated at

```
disp(datetime)
```

```
   19-Mar-2025 06:49:04
```

Define the source folder here:

```
main_folder =
"C:\Users\al3xm\Documents\_Local_Data\3.12.25_PAG_HILO_Beads\2.5ms_\ANM_XML"
```

```
main_folder =
"C:\Users\al3xm\Documents\_Local_Data\3.12.25_PAG_HILO_Beads\2.5ms_\ANM_XML"
```

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

```
paths =
"C:\Users\al3xm\Documents\_Local_Data\3.12.25_PAG_HILO_Beads\2.5ms_\ANM_XML\ANM_tSeries_BEADS_IN_PAG_388fps_11.xml"
```

```
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=0.002577 % 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 1721 tracks in the file.
found a total of 1721 tracks in the directory
Warning: Scaling factor active!
Warning: plotting only tracks longer than threshold length
Computing MSD of 843 tracks...   Done.
```

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

```
Fitting 843 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 843 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 = 0.8821
```

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

```
The diffusion coefficient is D=0.00016526microns^2/seconds
```

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

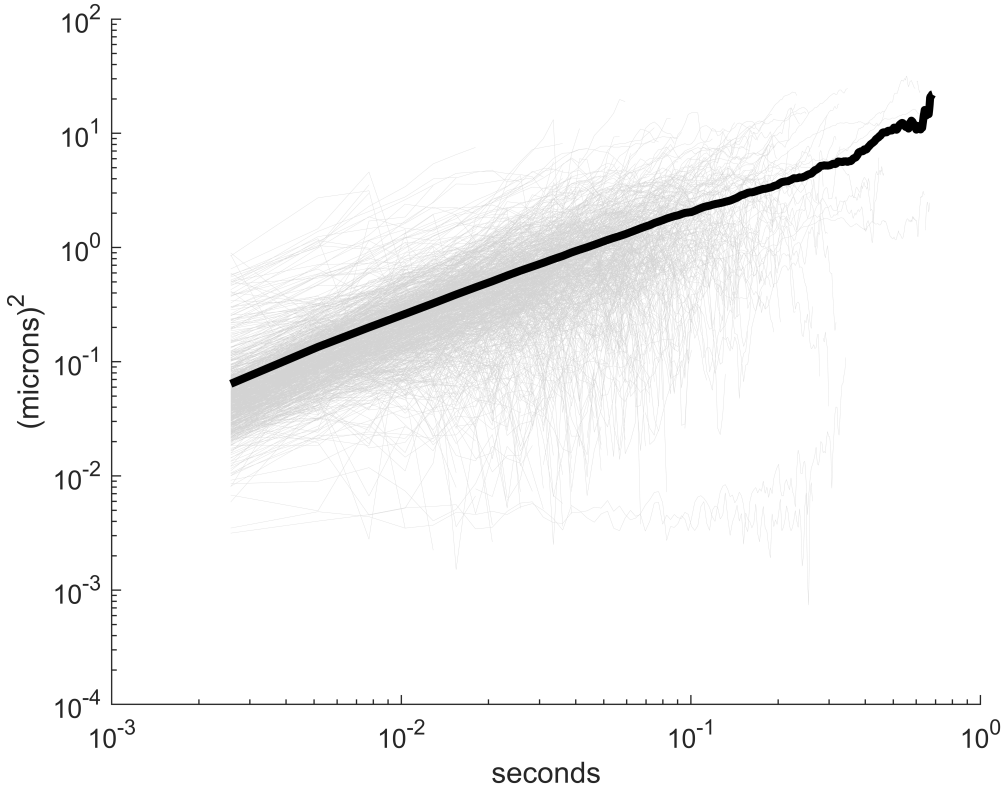
```
The time scaling factor is gamma=0.88214.
```

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

```
D_expect=-1;
fprintf(strcat('excpected diffusion coefficient: ', string(D_expect),' (',
SpaceUnits, ')^2/', TimeUnits))
```

```
excpected diffusion coefficient:-1 (microns)^2/seconds
```

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

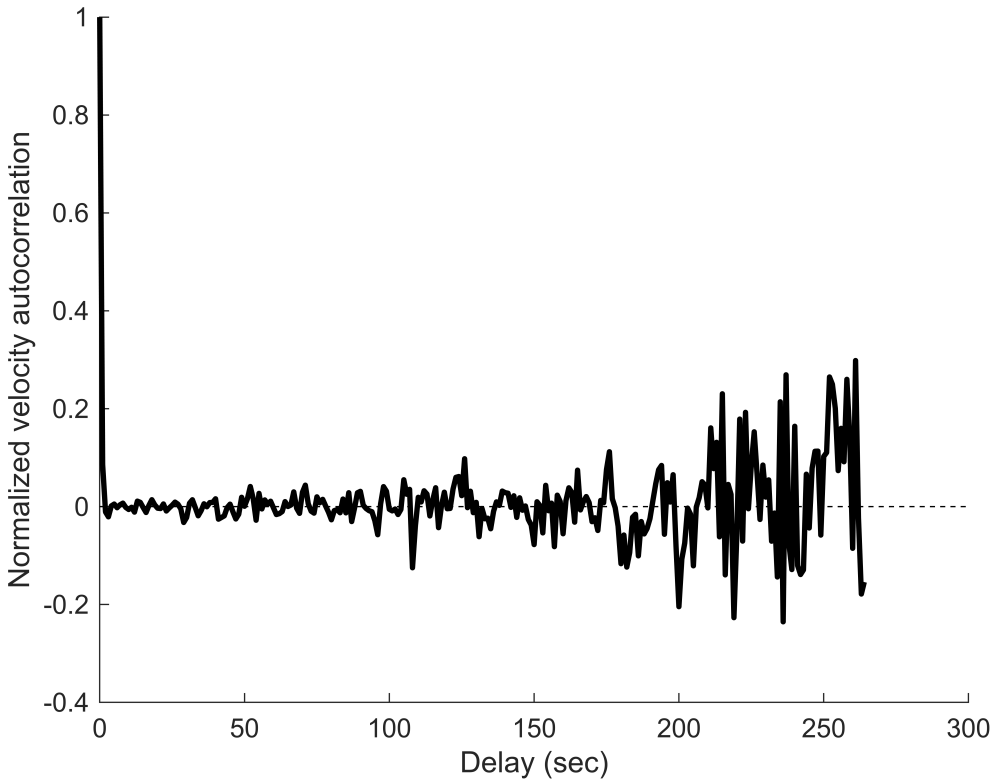



```
ans = 266×5
         0         0         0  434.2391         0
    0.0026    0.0641    0.0661  397.9462   -0.0103
    0.0052    0.1348    0.1214  356.9355   -0.0206
    0.0077    0.1999    0.1686  334.0387   -0.0309
    0.0103    0.2624    0.2053  315.4846   -0.0412
    0.0129    0.3246    0.2629  299.8320   -0.0515
    0.0155    0.3880    0.3140  284.1937   -0.0618
    0.0180    0.4484    0.3425  271.0939   -0.0722
    0.0206    0.5067    0.3742  257.6061   -0.0825
    0.0232    0.5684    0.4237  245.1368   -0.0928
       ⋮
```

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

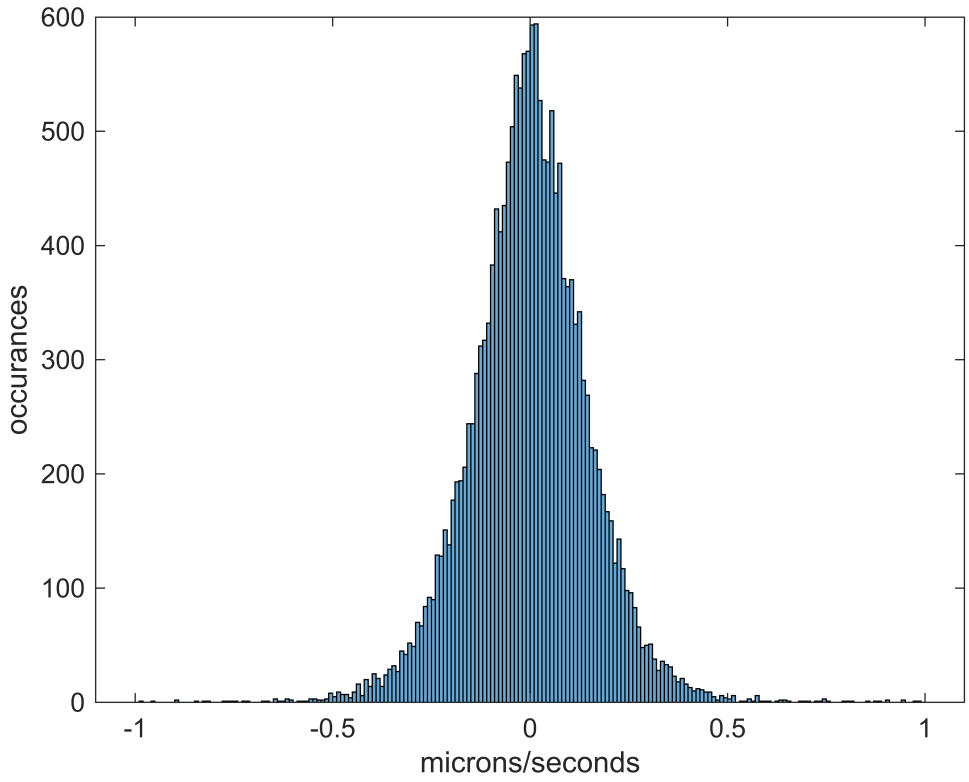
```
figure;
ma1.plotMeanVCorr;
```

```
Computing velocity autocorrelation of 843 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
```

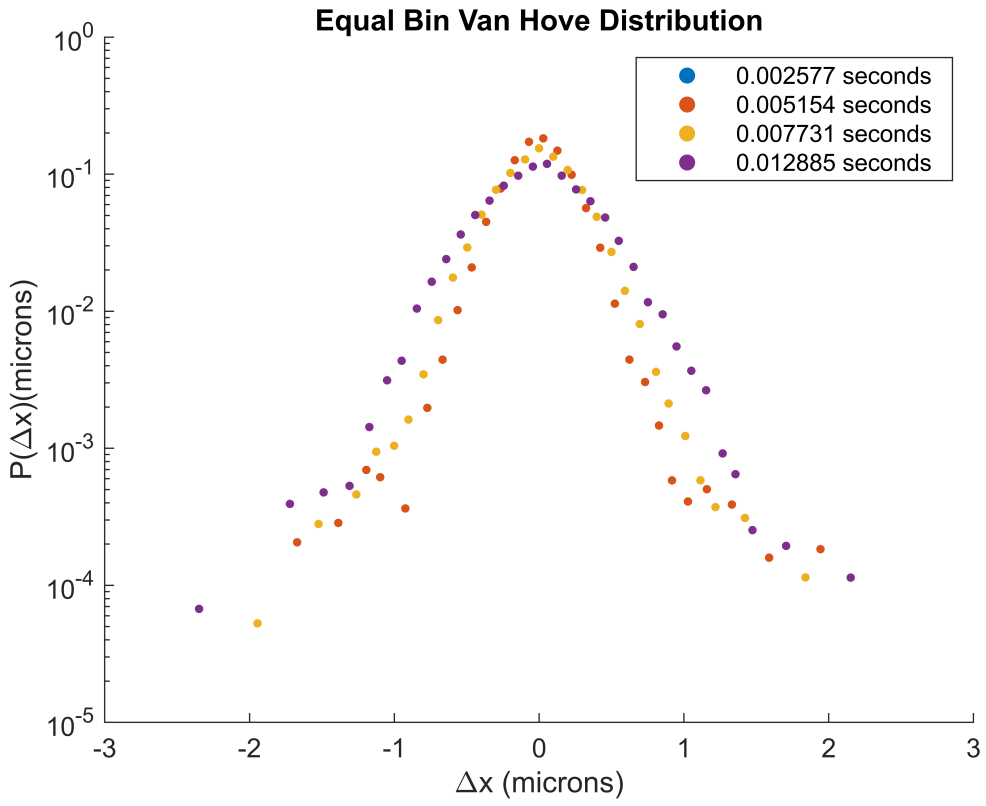
```
BinSize = 10
```

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

```
MinBin = 0.1000
```

```
VanHoveStats= 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
```




```
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: 0.1000
```

Save the data to the SuperStructs Github folder.

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

```
ans =
'C:\Users\al3xm\Documents\GitHub\HISTia\SuperStructs\Analysis of 605nm Qdots imaged at 388fps on Hestia'
```