Analysis of 605nm Qdots imaged at 200fps on Hestia

Name this data analysis run

```
monkier = 'Analysis of 605nm Qdots imaged at 200fps on Hestia';
disp(monkier)
```

```
Analysis of 605nm Qdots imaged at 200fps on Hestia
```

This Report generated at

```
disp(datetime)
```

```
   19-Mar-2025 06:51:24
```

Define the source folder here:

```
main_folder = "C:\Users\al3xm\Documents\_Local_Data\3.12.25_PAG_HILO_Beads\5ms"
```

```
main_folder =
"C:\Users\al3xm\Documents\_Local_Data\3.12.25_PAG_HILO_Beads\5ms"
```

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

```
paths =
"C:\Users\al3xm\Documents\_Local_Data\3.12.25_PAG_HILO_Beads\5ms\ANM_tSeries_BEADS_IN_PAG_200fps_11_Tracks.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=0.005 % 1/(fps)
```

```
dt = 0.0050
```

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 1815 tracks in the file.
found a total of 1815 tracks in the directory
Warning: Scaling factor active!
Warning: plotting only tracks longer than threshold length
Computing MSD of 866 tracks...   Done.
```

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

```
Fitting 866 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 866 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 = 0.8742
```

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

```
The diffusion coefficient is D=0.00051437microns^2/seconds
```

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

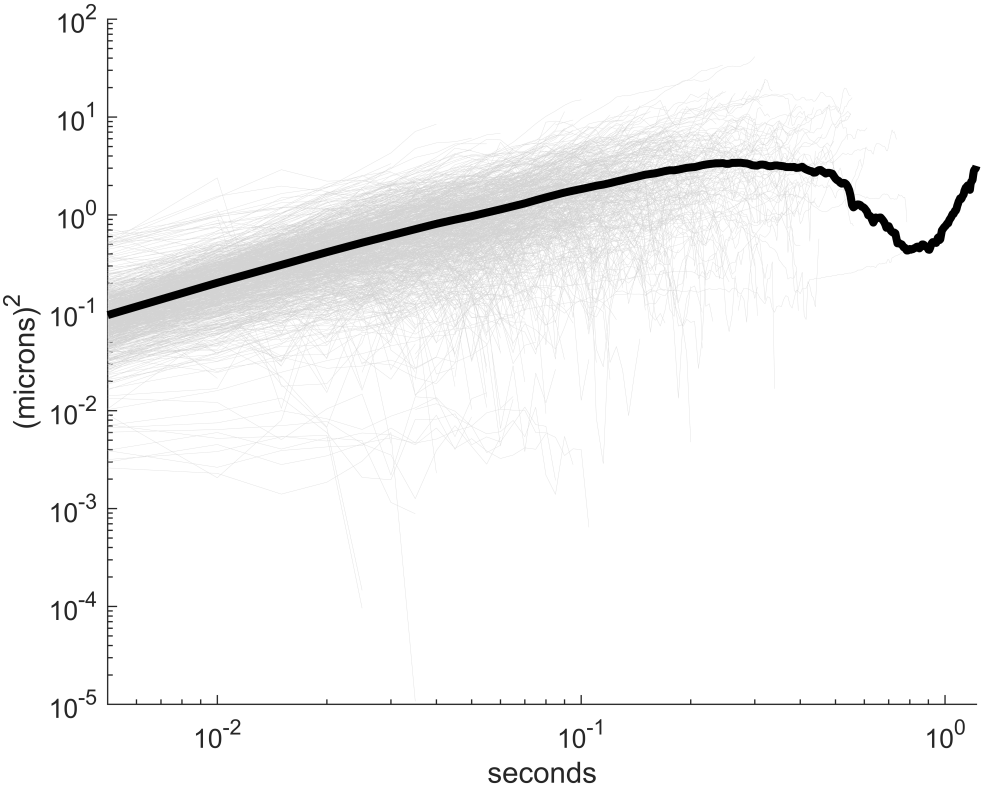
```
The time scaling factor is gamma=0.87419.
```

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)
```

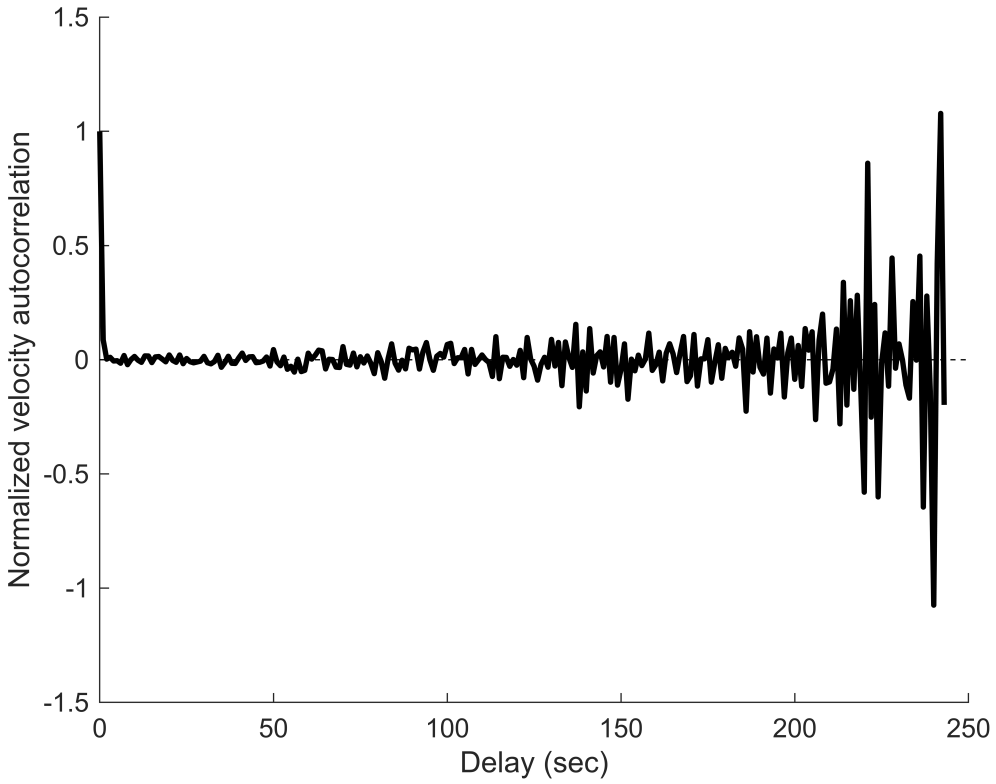



```
ans = 245×5
         0         0         0  480.7373         0
    0.0050    0.0948    0.0652  441.7143   -0.0200
    0.0100    0.2026    0.1194  396.5231   -0.0400
    0.0150    0.3086    0.1798  365.1914   -0.0600
    0.0200    0.4162    0.2413  341.7072   -0.0800
    0.0250    0.5201    0.3082  323.8580   -0.1000
    0.0300    0.6180    0.3739  306.3244   -0.1200
    0.0350    0.7148    0.4456  287.8153   -0.1400
    0.0400    0.8108    0.5108  272.4297   -0.1600
    0.0450    0.8945    0.5767  260.8137   -0.1800
      ⋮
```

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

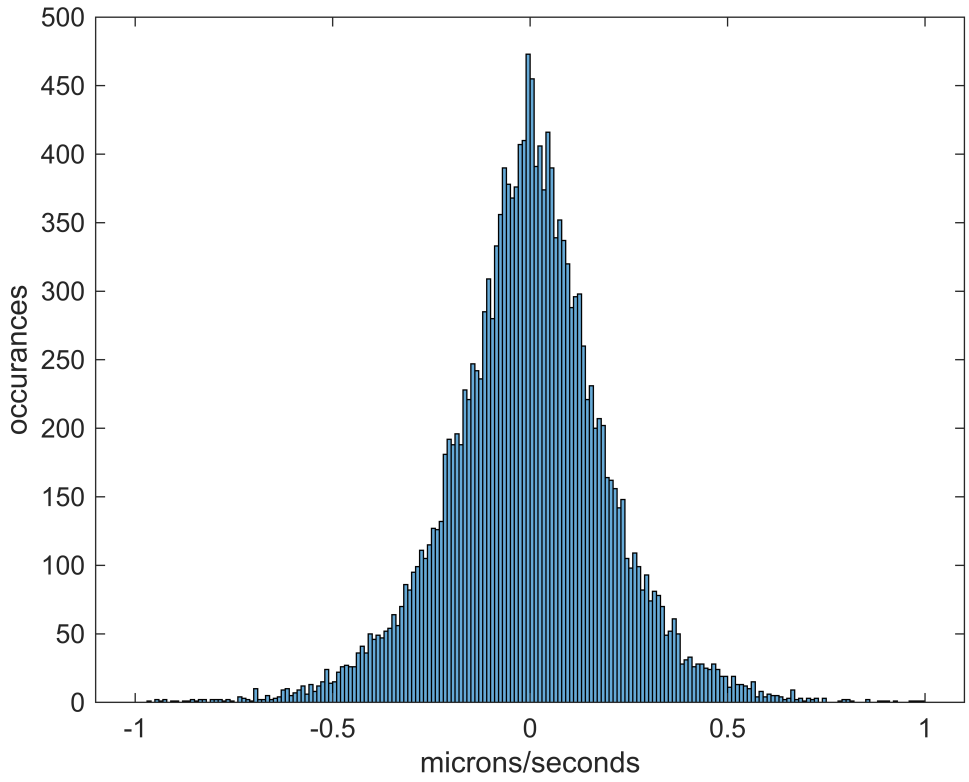
```
figure;
ma1.plotMeanVCorr;
```

```
Computing velocity autocorrelation of 866 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
```

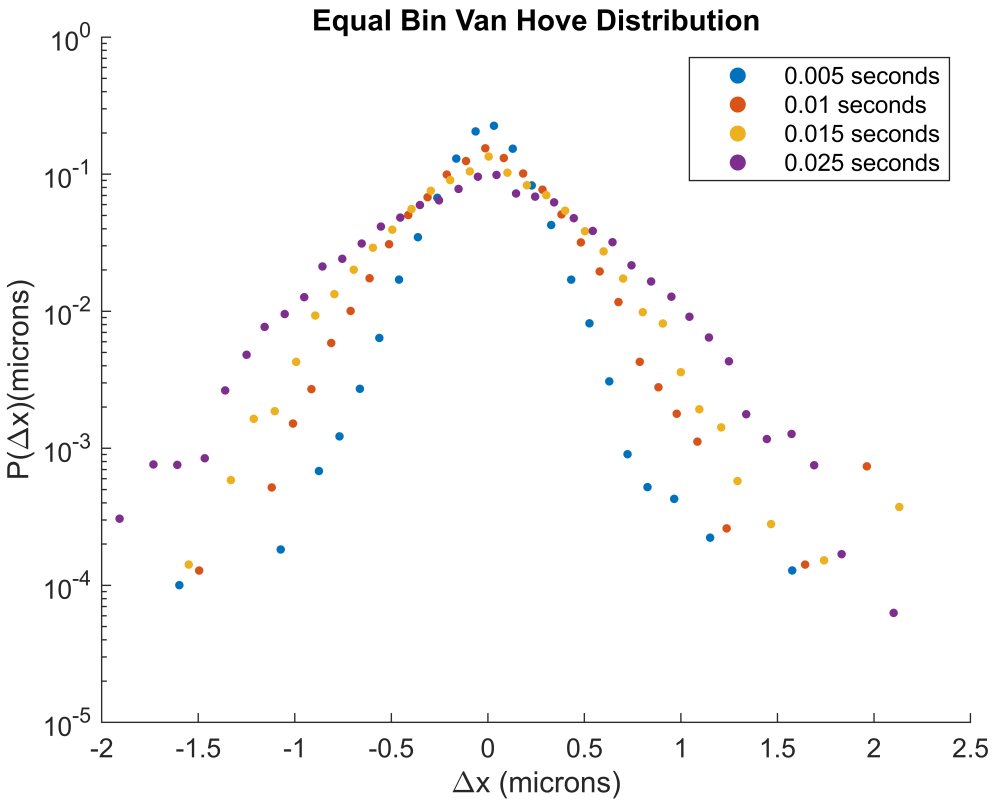
```
BinSize = 10
```

```
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: 0.0050
           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 200fps on Hestia2025-03-19.00
```