```
%% Set up folders and experimental conditions
folder=(['/Users/']);
savefolder = (['/Users/']);%not into "folder" !!!

%% load calibration file
%calibration=importdata('/Users/calibration.mat');

pixelsize=0.178; %pixelsize in µm
max_linking_distance=0.5; %in µm
max_gap_closing=1; % in frames
t=0.027; %total time per frame in sec
minimumtracklength=40; %in frames


%% call particletracker3D to generate tracks from Quickpalm reads

mkdir(savefolder);
subdirs = dir(folder);

SPACE_UNITS = 'µm';
TIME_UNITS = 's';
trackslists=[];
filteredtrackslists=[];

for i=1:length(subdirs)
    if isempty(strfind(subdirs(i).name,'.'))
       foldernames{i}=subdirs(i).name;
       resultstable=[folder '/' subdirs(i).name '/' 'Particles
Table.txt'];
       Channel1=[folder '/' subdirs(i).name '/' 'C2.tif'];
       currentfolder=[folder '/' subdirs(i).name '/'];
       [filteredtrackslist,
tracks]=particletracker3D_astigmatism_trackplotter(pixelsize,calibrati
on,resultstable,max_linking_distance,max_gap_closing,t,minimumtracklen
gth,Channel1,currentfolder);
       trackslists=[trackslists; tracks];
       filteredtrackslists=[filteredtrackslists; filteredtrackslist];
     end
end

ma = msdanalyzer(3, SPACE_UNITS, TIME_UNITS);
ma = ma.addAll(filteredtrackslists);

% Display the results
%% Plot the tracks
figure
ma.plotTracks;
ma.labelPlotTracks;
set(gca,'YDir','reverse');%restore orientation as in image
saveas(gca,[savefolder, '/' 'tracks' '.fig'])
```

```
%% Compute and plot the MSDs
ma = ma.computeMSD;
ma.msd
figure
ma.plotMSD;
saveas(gca,[savefolder, '/' 'MSDs' '.fig'])

%% Compute and plot the mean MSD
figure
ma.plotMeanMSD(gca, true)
mmsd = ma.getMeanMSD;
t = mmsd(:,1);
x = mmsd(:,2);
dx = mmsd(:,3) ./ sqrt(mmsd(:,4));
errorbar(t, x, dx, 'k')
saveas(gca,[savefolder, '/' 'Mean_MSDs' '.fig'])

%% fitting the MeanMSD curve
[fo, gof] = ma.fitMeanMSD(0.9);
plot(fo)
ma.labelPlotMSD;
legend off

%% fitting D from the individual MSD curves

ma = ma.fitMSD;
good_enough_fit = ma.lfit.r2fit > 0.8;
Dmean = mean( ma.lfit.a(good_enough_fit) ) / 2 / ma.n_dim;
Dstd  =  std( ma.lfit.a(good_enough_fit) ) / 2 / ma.n_dim;

fprintf('Estimation of the diffusion coefficient from linear fit of
the MSD curves:\n')
fprintf('D = %.3g ± %.3g (mean ± std, N = %d)\n', ...
    Dmean, Dstd, sum(good_enough_fit));


%% determine alpha

ma = ma.fitLogLogMSD(0.2);
ma.loglogfit
mean(ma.loglogfit.alpha)

r2fits = ma.loglogfit.r2fit;
alphas = ma.loglogfit.alpha;


%set fitting goodness cutoff
R2LIMIT = 0.8;
```

```
% Remove bad fits
bad_fits = r2fits < R2LIMIT;
fprintf('Keeping %d fits (R2 > %.2f).\n', sum(~bad_fits), R2LIMIT);
alphas(bad_fits) = [];

% T-test
[htest, pval] = ttest(alphas, 1, 0.05, 'left');

if ~htest
    [htest, pval] = ttest(alphas, 1, 0.05);
end

% Prepare string
str = { [ '\alpha = ' sprintf('%.2f ± %.2f (mean ± std, N = %d)',
mean(alphas), std(alphas), numel(alphas)) ] };

if htest
    str{2} = sprintf('Significantly below 1, with p = %.2g', pval);
else
    str{2} = sprintf('Not significantly differend from 1, with p = %.
2g', pval);
end

figure
hist(alphas);
box off
xlabel('\alpha')
ylabel('#')

yl = ylim(gca);
xl = xlim(gca);
text(xl(2), yl(2)+2, str, ...
    'HorizontalAlignment', 'right', ...
    'VerticalAlignment', 'top', ...
    'FontSize', 16)
title('\alpha values distribution', ...
    'FontSize', 20)
ylim([0 yl(2)+2])
saveas(gca,[savefolder, '/' 'alphas' '.fig'])


%% Velocity analysis
figure
v = ma.getVelocities;
V = vertcat( v{:} );

hist(V(:, 2:end), 50) % we don't want to include the time in the
histogram
box off
xlabel([ 'Velocity (' SPACE_UNITS '/' TIME_UNITS ')' ])
```

```
ylabel('#')
saveas(gca,[savefolder, '/' 'velocity' '.fig'])

mean_velocity = mean(V(:,2:end));
std_velocity = std(V(:,2:end));


%% save ma file

save([savefolder, '/data.mat']);
```