```
function
[filteredtrackslist,FN_x_y_WH]=calibrationcurve_trackplotter(pixelsize
,resultstable,max_linking_distance,max_gap_closing,zspacing,minimumtra
cklength,Channel1,currentfolder)


%% Import Quickpalm results and perform track linking


Results=importdata([resultstable]);
n_dim = 2;
debug = true;

%sorting xy and width height coordinates from QuickPALM Results.txt
and putting into new
%array

FN=(Results.data(:,end));
x=(Results.data(:,5));
y=(Results.data(:,6));
WidthHeight=(Results.data(:,14));

[FN_sorted,sortIdx] = sort(FN);
x_sorted = x(sortIdx);
y_sorted = y(sortIdx);
WidthHeight_sorted = WidthHeight(sortIdx);

%convert x/y to µm
x_sorted=x_sorted/1000;
y_sorted=y_sorted/1000;

FN_x_y=[FN_sorted x_sorted y_sorted];
FN_x_y_WH=[FN_sorted x_sorted y_sorted WidthHeight_sorted];

%generating "points" cell with each frame having its own array
clear points
points{1,max(FN_x_y(:,1))}=[];
points=points';
for i=1:length(FN_x_y)
    points{FN_x_y(i,1)}=[points{FN_x_y(i,1)}; FN_x_y(i,2:3)];
end

%Fill empty arrays with zeros. (As QuickPALm sometimes cannot localize
any
%particle in a frame) but Simpletracker needs values in each frame)

for j=1:length(points)
    if (isempty(points{j,1}))==1
       points{j,1}=[0,0];
    end
```

```
end

%generating "pointsWH" cell with each frame having its own array
clear pointsWH
pointsWH{1,max(FN_x_y_WH(:,1))}=[];
pointsWH=pointsWH';
for i=1:length(FN_x_y_WH)
    pointsWH{FN_x_y_WH(i,1)}=[pointsWH{FN_x_y_WH(i,1)}; FN_x_y_WH(i,
1:4)];
end

%Fill empty arrays with zeros. (As QuickPALm sometimes cannot localize
any
%particle in a frame) but Simpletracker needs values in each frame)

for j=1:length(pointsWH)
    if (isempty(pointsWH{j,1}))==1
       pointsWH{j,1}=[0,0,0,0];
       display('zero added')
    end
end

%track linking

[ tracks adjacency_tracks ] = simpletracker(points,...
    'MaxLinkingDistance', max_linking_distance, ...
    'MaxGapClosing', max_gap_closing, ...
    'Debug', debug);

%concatenate pointsWH
allpoints=vertcat(pointsWH{:});


%lookup the point numbers from "adjacency_tracks" in "FN_x_y_WH" and
%combine them in a cell array "tracks"
clear tracks
tracks=cell(length(adjacency_tracks),1);

for i=1:length(adjacency_tracks)
    trackpointlist=adjacency_tracks{i,1};


    for k=1:length(trackpointlist)
        tracks{i,1}=[tracks{i,1}; (allpoints((trackpointlist(k,1)),
1)*zspacing), (allpoints((trackpointlist(k,1)),2:4))];

    end


end
```

```
%filter out tracks made of empty frames (zero-tracks)

trackslist={};
counter=0;
for i=1:length(tracks)

    if sum( sum(tracks{i,1}))==0;
    display('found and deleted zero track')
    else
    counter=counter+1;
    trackslist{counter,1}=tracks{i,1};
    end

end

%filter tracks shorter than minimumtracklength

clear i
filteredtrackslist=[];

for i=1:length(tracks)
    if length(tracks{i,1})>= minimumtracklength
                filteredtrackslist=[filteredtrackslist; tracks(i,1)];
    end

end

%% track plotting

%% get metadata for channel 1
InfoImageCh1=imfinfo(Channel1);
mImageCh1=InfoImageCh1(1).Width;
nImageCh1=InfoImageCh1(1).Height;
NumberImagesCh1=length(InfoImageCh1);

framenumber=length(InfoImageCh1);

%% import channel 1

clear i
FinalImageCh1=zeros(nImageCh1,mImageCh1,NumberImagesCh1,'uint16');
for i=1:NumberImagesCh1
   FinalImageCh1(:,:,i)=imread(Channel1,'Index',i);
end

%% prepare tracks to be plotted into image

%concatanate filteredtrackslist
```

```
filteredtracksperframe2 = vertcat( filteredtrackslist{:} );

framet = filteredtracksperframe2(:,1);
x = filteredtracksperframe2(:,2);
y = filteredtracksperframe2(:,3);

%convert x,y from pixels to µm and round to full pixels
x=round(x/pixelsize);
y=round(y/pixelsize);

%sort by t
[t_sorted,sortIdx] = sort(framet);
x_sorted = x(sortIdx);
y_sorted = y(sortIdx);

%convert t to FN
FN_sorted = round(t_sorted/zspacing);%round is needed here to make
sure integers are the result

trackperframesorted=[FN_sorted, x_sorted, y_sorted];
trackperframesorted=uint16(trackperframesorted);

%generate new cell structure with each frame having its own arrray

Ch1points{(framenumber),1}=[];

for i=1:length(trackperframesorted)
    Ch1points{trackperframesorted(i,1)}
=[Ch1points{trackperframesorted(i,1)}; trackperframesorted(i,1:3)];
end


%Fill empty arrays with zeros.
for j=1:length(Ch1points)
    if (isempty(Ch1points{j,1}))==1
       Ch1points{j,1}=[0,0,0];
    end
end



%% plot tracks into file

clear i
clear k
Modimage=zeros((mImageCh1+8),(nImageCh1+8),framenumber);
Modimage=uint16(Modimage);
trackimage=zeros((mImageCh1+8),(nImageCh1+8));

for     i=1:length(Ch1points)
```

```
        if  sum(Ch1points{i,1})>0
            Image=zeros((nImageCh1+8),(mImageCh1+8)); %make image of
size prelimmage + 8 in each dimension so that white square fits

            %annotate next two lines out if you want tracks on black
background

%             prelimImage=(FinalImageCh1(:,:,i));%extract picture i
%             Image(5:(nImageCh1+4),5:(mImageCh1+4))=prelimImage; %put
prelimimage into Image


            Image=imrotate(Image, 270); %rotate array 90 degrees
clockwise
            Image=flipdim(Image,2); %flip array horizontally as
Quickpalm flips the coordinates



            for k=1:(length (Ch1points{i,1}(:,1)))

                xpix=(Ch1points{i,1}(k,2))+4;
                ypix=(Ch1points{i,1}(k,3))+4;

                trackimage(xpix,ypix)=65536;%makes white dot at
particle position

                %draw white square around particle position
                Image((xpix-4):(xpix+4),ypix-4)=65536;
                Image((xpix-4):(xpix+4),ypix+4)=65536;
                Image(xpix-4,(ypix-4):(ypix+4))=65536;
                Image(xpix+4,(ypix-4):(ypix+4))=65536;

                Image=Image+trackimage;

                %draw white pixel at particle position
                Image(xpix,ypix-4)=65536;

            end

            Modimage(:,:,i)=[Image];

        else

            %if tracks on black backgound are ok:
            Modimage(:,:,i)=zeros((mImageCh1+8),(nImageCh1+8));

            %if plotting tarcks into image then code is needed here
that
```

```
            %takes ith image, flips and turns it and inserts it into
bigger
            %image wit dimension+8

        end


end

%restore orientation of image

Modimage=imrotate(Modimage, 90); %rotate array 90 degrees
counterclockwise
Modimage=flipdim(Modimage,1); %flip array vertically

%crop back Modimage to original size

Modimage=Modimage(5:(nImageCh1+4),5:(mImageCh1+4),1:framenumber);

%write final image stack

t = Tiff([currentfolder, '/tracks.tif'],'w');

for i=1:size(Modimage,3)


t = Tiff([currentfolder, '//tracks.tif'],'a');

tagstruct.ImageLength = nImageCh1;
tagstruct.ImageWidth = mImageCh1;
tagstruct.Photometric = Tiff.Photometric.MinIsBlack;
tagstruct.SampleFormat = Tiff.SampleFormat.UInt;
tagstruct.Compression = Tiff.Compression.None;
tagstruct.BitsPerSample = 16;
tagstruct.SamplesPerPixel = 1;
tagstruct.RowsPerStrip = 1;
tagstruct.PlanarConfiguration = Tiff.PlanarConfiguration.Chunky;
tagstruct.Software = 'MATLAB';
t.setTag(tagstruct);


t.write(Modimage(:, :, i));

t.close();

end


end
```