```
function linkTrajSegments(image_label,start_frame,end_frame,spot_results,data_set_label)
%
% Created by Isabel Llorente-Garcia, Sept. 2011.
% If you use this code please acknowledge Isabel Llorente-Garcia in your
% publications.
%
% This function links and groups all found trajectory segments (of bright fluorescent spots)
% and overlays fully connected trajectories on image sequence in a video.
% Trajectory data results are exported to an .xls file.
%
% INPUTS:
% image_label: string such as '513', '490', etc... which corresponds to a certain label of image sequence.
% start_frame and end_frames are the frames through which the loop runs to plot the result trajectories.
% They should be the same as previously used to produce spot_results using FindTrajects.m.
% spot_results: parameters (spot_results{1}) and calculated trajectory segments (spot_results{2}). It is the output of↙
function FindTrajects.m
% (it is a cell array with two elements. The second one is a structure array containing all segment results).
% data_set_label: is a user-defined label that will form part of the name of the output excel file. E.g.: 'ATPase-GFP'.
%
% Example of how to call this function:
% linkTrajSegments('490',100,110,s1,'ATPase-GFP'),
% where s1 =FindTrajects('490',100,110) has been previously calculated;
% Takes image sequence 490 within the current folder,
% and plots the full trajectories that result from linking the trajectory segments in s1,
% overlaid on the image sequence, for frames 100 to 110.
% -------------------------------------------------------
% Note: for reading .sif files you need 'read_sif_data_direct.m',
% 'GetAndorSifSize', etc. which are currently in path:
% 'C:\Isabel\myMatlabFiles\IO_Input\'.
%
% NOTE: before running this function you should move into the directory which
% contains the image sequence data (labelled by 'image_label').


%% DEFINITIONS and PARAMETERS:
%
% define data (.sif image) directory:
```

```matlab
% dir_data = 'Z:\Leake\Heiko Data\';
% dir_data = 'C:\Isabel\ExperimData\HeikoData\';
% print data directory on command window to guide user:
disp(' ') % empty line
disp(['The data directory (image sequences) is: ',cd])

disp(' ') % empty line
disp(['The start frame for plotting trajectories is ',num2str(start_frame)]) % start_frame is an input.
disp(['The end frame for plotting trajectories is ',num2str(end_frame)]) % end_frame is an input.


% I have saved all parameter values in the file "paramsForLinkTrajSegments.m"
% in the current directory. Just calling the name of that file loads the
% parameter values into the workspace:
paramsForLinkTrajSegments
% In this way, we can save different parameter sets for different data sets
% in an easy manner and run several Matlabs at the same time working with
% different parameter sets.


% % Use a very large number (larger than image size in pixels) for rejected asignments:
% rej = 10000;
% params.rej = rej; % Save parameters to results as structure "params".
%
% % PARAMETERS for linking trajectory segments:
% % For linking end spot in one trajectory with start spot in another trajectory:
% d_01_max = 5; % max distance in pixels between spot centres.(5)
% Iratio_01_min = 0.2; % min ratio of total spot intensities (after bgnd subtraction).(0.2)
% Iratio_01_max = 4; % max ratio of total spot intensities (after bgnd subtraction). (Large enough to account for blinking,↙
note that Iratio is for frame k-1/frame k.)(3)
% SigmaRatio_01_min = 0.5; % min ratio of spot widths (sigma of Gaussian fit).(0.5)
% SigmaRatio_01_max = 2; % max ratio of spot width (sigma of Gaussian fit). (2)
% Frames_away_max = 2; % max separation in frames (i.e., prop to time) for trajectory segments to be linked.(default = 2).↙
Write 1 if you want no jumps (no frame jumps) in the segment linking.
% % At most we skip one frame when Frames_away_max = 2.
% % Save parameters to results as structure "params":
% params.d_01_max = d_01_max;
% params.Iratio_01_min = Iratio_01_min;
% params.Iratio_01_max = Iratio_01_max;
% params.SigmaRatio_01_min = SigmaRatio_01_min;
% params.SigmaRatio_01_max = SigmaRatio_01_max;
% params.Frames_away_max = Frames_away_max;
%

%% Extract segment (small trajectory) results from "spot_results" input:

params_for_FindTrajects = spot_results{1}; % parameters used in function FindTrajects.m to obtain segments.
traj_results = spot_results{2}; % Results of segments, obtained by function FindTrajects.m.


%% list of colours for plotting trajectories:
% colors are in this order: red, green, blue, yellow, pink, lighter blue,
% darker red, darker green, darker blue, mustard, darker pink, light blue,
% light orange, light greenish, darker green, salmon, another light blue, very light green,
% lighter grey, light gray, light yellow, light pink, light blue, orange
% the length of this list determines the total no. of trajectories we can plot (7200 now).
color_list_0 = [1 0 0;     0 1 0;   0 0 1;     1 1 0;     1 0 1;     0 1 1;...
    0.8 0 0; 0 0.8 0; 0 0 0.8; 0.8 0.8 0; 0.8 0 0.8; 0 0.8 0.8;...
    0.95 0.72 0.4; 0.86 0.95 0.6; 0.4 0.8 0.4; 0.9 0.6 0.5; 0.4 0.8 0.7; 0.85 0.92 0.99;...
    0.9 0.9 0.9; 0.7 0.7 0.7; 1 1 0.5; 1 0.5 1; 0.5 1 1; 1 0.6 0.2];
% Make of class single first to save memory:
color_list = repmat(single(color_list_0),300,1); % replicate copies of the matrix color_list_0.

% ------------
%% Error control: if the number of trajectories to be plotted is larger than
% the number of colours in color_list, exit function:
if max([traj_results.TrajNumber])> length(color_list)
    disp('') % empty line.
    disp('!!')
    disp('ATTENTION!!:  number of trajectories to be plotted exceeds length of list of colours to plot...')
    disp('')
    disp('Exiting function.')
    disp('No excel file of trajectory results was saved.')
    disp('!!')
    disp('')
    return
end
% ------------

%% Read in the .sif image data:
%
% Read image-sequence file:
[numFrames frame_Ysize frame_Xsize image_data image_path] = extract_image_sequence_data(image_label);
% See "extract_image_sequence_data.m".
% numFrames is the number of frames in the image sequence.
% To get frame number "p" do: image_data(p).frame_data.
% Frame dimensions are frame_Ysize and frame_Xsize.
% --------------------------------------------------------------


%% SORT and ARRANGE TRAJECTORY DATA, step 1:
% Convert structure resulting from function FindTrajects.m into useful
% data arranged in a large matrix (spot after spot).

max_no_spots = size(traj_results,2); % max. number of found spot centres in each frame.

data_to_export_labels = fieldnames(traj_results)'; % Row vector with the fieldnames of all found spots.
%↙
('CentreX','CentreY','IspTot','rsqFit','SigmaFit','I0Fit','BgNoiseStd','IbgAvg','IbgTot','SNR','IinnerTot','ClipFlag','TrajNumber↙
','FrameNumber','SpotNumber').

% Make of class single to save memory space:
all_data_to_export = single([]); % empty, to be filled in (original data, all spots).
data_to_export = single([]); % empty, to be filled in (only spots that have been linked into trajectories will be saved to↙
this variable).

traj_column = strmatch('TrajNumber',data_to_export_labels); % number of column containing trajectory numbers (used later to↙
sort traj data by TrajNumber).
x_column = strmatch('CentreX',data_to_export_labels); % number of column containing CentreX position (used later).
y_column = strmatch('CentreY',data_to_export_labels); % number of column containing CentreY position (used later).
IspTot_column = strmatch('IspTot',data_to_export_labels); % number of column containing IspTot (used later).
```

```
Sigma_column = strmatch('SigmaFit',data_to_export_labels); % number of column containing SigmaFit (used later).
Frame_column = strmatch('FrameNumber',data_to_export_labels); % number of column containing Frame number (used later).

% Loop through selected frames:
for k = start_frame:end_frame

%     frame = image_data(k).frame_data; % extract frame data which is stored in field 'frame_data'.
%     frame = double(frame);
%     imshow(frame,[],'Border','tight','InitialMagnification',150); % show image scaled between its min and max values ([]).
%     hold on;
%     pause(0.5);
%
%     disp(['frame number: ',num2str(k)]) % print frame number to Command Window.
%
%     % Plot result trajectories:
    for q = 1:max_no_spots % loop through found spots on each frame:

        % traj_results(k,q) is a structure containing the found spot
        % characteristics, including fields 'CentreX', 'CentreY' and
        % 'TrajNumber'.
        tr = traj_results(k,q).TrajNumber; % trajectory number corresponding to found spot q on frame k.

        if ~isempty(traj_results(k,q).FrameNumber) % if the spot is not empty, save it to all_data_to_export.
            if isempty(traj_results(k,q).TrajNumber) % if TrajNumber is empty fill it up with a 0 or dimensions in next step will not match.
                traj_results(k,q).TrajNumber = 0;
            end
            all_data_to_export = [all_data_to_export; single(cell2mat(struct2cell(traj_results(k,q)))')]; % append row with the spot data.
        end

        if  tr > 0  % only plot trajectory numbers > 0 (TrajNumber=0 is for loose spots):
            % Plot each trajectory in a different colour:
%             plot(traj_results(k,q).CentreX,traj_results(k,q).CentreY,'o','Color',color_list(tr,:),'MarkerSize',10)
            % Construct trajectory data to save (convert structure to matrix):
            data_to_export = [data_to_export; single(cell2mat(struct2cell(traj_results(k,q)))')]; % append row with the spot data.
            end
        end
%       pause(0.2);
%       hold off;

end  % loop through selected frames.
% ----------------------------------


%% SORT and ARRANGE TRAJECTORY DATA, step 2:
% Sort spot data by trajectory number and separate trajectories:

% % Unsorted trajectory data:
% data_to_export_unsorted = [data_to_export_labels; num2cell(data_to_export)]; % Add as a first row the fieldnames (labels↙
for each column).

% Sort spots data by trajectory number:
if ~isempty(data_to_export)
    data_to_export_sorted = sortrows(data_to_export,traj_column); % sort data by trajectory number. This is of class single↙
by assignment.
else
    disp('!!')
    disp('WARNING: spots found were not linked into trajectories. No trajectories of more than one point were found.')
    disp('No excel file of trajectory results was saved.')
    disp('!!')
    return
end

trajs_max = max([traj_results.TrajNumber]); % number of trajectories to be plotted/saved.

% trajs_max

% Make cell array in which each element is a matrix corresponding to a given trajectory number:
separated_trajs = cell(1,trajs_max); % make cell array (row) of empty matrices. (Pre-alocate size for speed.)
i = 1; % index for rows of sorted trajectory data.
for tr = 1:trajs_max % loop through all trajectory numbers:
    j = 1; % index for rows of separated trajectory data.
    while data_to_export_sorted(i,traj_column) == tr
        separated_trajs{tr}(j,:) = data_to_export_sorted(i,:); % fill up row by row.
        if i == size(data_to_export_sorted,1) % if we reach end row of data_to_export_sorted.
            break
        end
        i = i+1;
        j = j+1;
    end
end
% The result separated_trajs is a cell array with as many matrices (of
% different sizes) as trajectories. E.g.: separated_trajs{tr} gives the
% data for trajectory "tr". E.g. separated_trajs{tr}(:,1) gives all the
% CentreX values (1st column) of trajectory number "tr".
% Note that the first row in each trajectory is reserved for adding column
% labels latter!!!
%--------------------------------


% Error control: if there is only one trajectory number (traj number 1 or 0 for loose spots):
if unique([traj_results.TrajNumber])<=1
    disp(' ') % empty line.
    disp('NOTE: there is only one trajectory in the input data. This trajectory has been sent to an excel file.')
    % Export initial sorted trajectory data (before linking trajectory segments):
    data_to_export_sorted = [data_to_export_labels; num2cell(data_to_export_sorted)]; % Add as a first row the fieldnames (labels for each column).
    output_filename = strcat(data_set_label,'_',image_label,'fullTrajs.xls'); % output .xls filename (before sorting data by trajectory number)
    xlswrite(output_filename,data_to_export_sorted); % write data to excelfile.
    % the previous data contains only spots which have been linked into a
    % trajectory.

    % Export also all (non-empty) spots in the input (traj_results) structure, even if they have not been linked into trajectories, i.e.,
    % export "all_data_to_export":
    all_data_to_export_2 = [data_to_export_labels; num2cell(all_data_to_export)]; % Add as a first row the fieldnames (labels for each column).
    output_filename = strcat(data_set_label,'_',image_label,'allspots.xls'); % output .xls filename (before sorting data by
```

```matlab
trajectory number)
    xlswrite(output_filename,all_data_to_export_2); % write data to excel file.

    return % exit function
end


%% LINKING TRAJECTORY SEGMENTS:
% Compare end point of a given trajectory with starting point of another
% one. Create matrices with results of pair-wise spatial distances, intensity ratios, sigma ratios and distance in frames.

% Create cell arrays with empty elements to pre-asign sizes:
A = single(zeros(trajs_max,trajs_max)); % create matrix of the right size, with all zeros, single precission.

d01 = num2cell(A); % Note: d01 is a cell array (matrix) but d_01 below is a scalar.
Iratio01 = num2cell(A); % Note: Iratio01 is a cell array (matrix) but Iratio_01 below is a scalar.
SigmaRatio01 = num2cell(A); % Note: SigmaRatio01 is a cell array (matrix) but SigmaRatio_01 below is a scalar.
FramesAway =  num2cell(A); % Note: FramesAway is a cell array (matrix) but Frames_away below is a scalar.
d01_best = num2cell(A); % Cell array which will contain only best asignment

clear A % remove variable from workspace

% d01 = cell(trajs_max,trajs_max); % Note: d01 is a cell array (matrix) but d_01 below is a scalar.
% Iratio01 = cell(trajs_max,trajs_max); % Note: Iratio01 is a cell array (matrix) but Iratio_01 below is a scalar.
% SigmaRatio01 = cell(trajs_max,trajs_max); % Note: SigmaRatio01 is a cell array (matrix) but SigmaRatio_01 below is a scalar.
% FramesAway =  cell(trajs_max,trajs_max); % Note: FramesAway is a cell array (matrix) but Frames_away below is a scalar.
%
% d01_best = cell(trajs_max,trajs_max); % Cell array which will contain only best asignments
% of pairs of trajectory segments after solving all possible competitions
% first within row and then within column in d01.

for qend = 1:trajs_max % loop though end points of all trajectories.
    %     qend
    for qstart = 1:trajs_max % loop though start points of all trajectories.
        % [qend qstart]
        d01_best{qend,qstart} = single(rej); % preliminarily fill up the best-assignment distance matrix with values rej.

        x_end = separated_trajs{qend}(end,x_column); % CentreX position for last spot in trajectory qend.
        y_end = separated_trajs{qend}(end,y_column); % CentreY position for last spot in trajectory qend.
        IspTot_end = separated_trajs{qend}(end,IspTot_column); % IspTot for last spot in trajectory qend.
        Sigma_end = separated_trajs{qend}(end,Sigma_column); % SigmaFit for last spot in trajectory qend.
        Frame_end = separated_trajs{qend}(end,Frame_column); % frame number for last spot in trajectory qend.

        x_start = separated_trajs{qstart}(1,x_column);   % CentreX position for first spot in trajectory qstart.
        y_start = separated_trajs{qstart}(1,y_column);   % CentreY position for first spot in trajectory qstart.
        IspTot_start = separated_trajs{qstart}(1,IspTot_column); % IspTot for first spot in trajectory qstart.
        Sigma_start = separated_trajs{qstart}(1,Sigma_column); % SigmaFit for first spot in trajectory qstart.
        Frame_start = separated_trajs{qstart}(1,Frame_column); % frame number for first spot in trajectory qstart.

        % d_01: distance between end spot in trajectory qend and start point in trajectory qstart:
        d_01 = sqrt((x_end-x_start)^2+(y_end-y_start)^2);
        % Iratio_01: ratio of intensities of spot centre in previous and current frames:
        Iratio_01 = IspTot_end/IspTot_start;
        % SigmaRatio_01: ratio of widths of spots (Gaussian fits) in previous and current frames:
        SigmaRatio_01 = Sigma_end/Sigma_start;
        % Distance in frames (proportional to time) between end spot in trajectory qend and start point in trajectory qstart:
        Frames_away = Frame_start-Frame_end; % should be >0 for linking segments (see later).

        %                         d_01
        %                         Iratio_01
        %                         SigmaRatio_01
        %                         Frames_away

        % Accept asignment (link trajectory segments) and save values only if start and end spots in
        % respective trajectories fulfill the following conditions:
        if d_01 < d_01_max && ...  % see PARAMETERS at start of this function.
                Iratio_01_min <= Iratio_01 && Iratio_01 <= Iratio_01_max && ...
                SigmaRatio_01_min <= SigmaRatio_01 && SigmaRatio_01 <= SigmaRatio_01_max && ...
                Frames_away > 0 && Frames_away <= Frames_away_max
            % Asign accepted values to cell array elements to store them:
            d01{qend,qstart} = d_01; % use {} for cell arrays.
            % Note that, by construction, cell array d01 has all elements in
            % and below its diagonal equal to rej. this is because
```

```
                % trajectories appear in the algorithm in order frame by frame,
                % so the end of a trajectory qend can never happen before the
                % beginning of a trajectory q start, for elements
                % d01{qend,qstart} with qend>=qstart, and therefore these
                % elements are all equal to rej.
                Iratio01{qend,qstart} = Iratio_01;
                SigmaRatio01{qend,qstart} = SigmaRatio_01;
                FramesAway{qend,qstart} = Frames_away;
            else % rejected asignments:
                d01{qend,qstart} = single(rej); % Use rej for asignments not accepted (images usually 512x512arrays, so rej pix↙
is an impossibly large distance, this is why it is chosen here).
                Iratio01{qend,qstart} = single(rej); % Use rej for asignments not accepted.
                SigmaRatio01{qend,qstart} = single(rej); % Use rej for rejected asignments.
                FramesAway{qend,qstart} = single(rej); % Use rej for rejected asignments.
            end
        end

        %                     d01
        %                     [d01{qend,:}]

        % Note that [d01{qend,:}] gives only non-empty elements of row qend
        % in the cell array d01 as a row vector, that's why we had to
        % give a numeric value rej to non-accepted asignments.

        % Note that if all asignments in previous step are rejected,
        % [d01{qend,:}] will be a list of rej values, and its minimum will
        % be rej.
        % If list of "linkable" segments, [d01{qend,:}] (row vector), has at least one accepted
        % asignment (at least one value smaller than rej):
        if min([d01{qend,:}]) ~= rej
            % if there is at least one accepted asignment in that row for trajectory segments:

            % Decide of all possible asignments of starting trajectory segments (qstart) which one is the best:
            % We take the best as the closest (minimum distance d01):

            % sort non-empty elements in row qend of d01 from smaller to larger distance:
            sorted_row = sort([d01{qend,:}]);
```

```matlab
        % For each, check column-wise:

        % for the minimum distance (if we accept it, we do nothing else):
        if sorted_row(1) < rej
            qstart_chosen = find( [d01{qend,:}] == sorted_row(1) ); % find position of pair-wise distance 1 in the row (find which column).

            %             qstart_chosen

            % Check column-wise in that column in d01 to avoid asigning a traj segment start qstart to a traj segment end qend that had already
            % had a traj segment asigned to it which might be at a shorter distance than the current one.

            %                                  [d01{:,qstart_chosen}] % chosen column of d01 matrix of distances.

            % If the found distance in that column is not the minimum one we don't proceed with the asigment, and we move on to the next smaller distance in sorted_row:
            if qend ~= find([d01{:,qstart_chosen}] == min([d01{:,qstart_chosen}]));
                d01_best{qend,qstart_chosen} = single(rej+0.1); % we re-set that distance to rej+0.1 (in case we need to distinguish these cases from cases rej later on).
                % if rejected, it moves on to next element in sorted_row
                % (go inside loop ii):

                if length(sorted_row)>=2
                    for ii = 2:length(sorted_row)
                        if sorted_row(ii) < rej
                            qstart_chosen = find( [d01{qend,:}] == sorted_row(ii) ); % find position of pair-wise distance ii in the row (find which column).

                            %                             qstart_chosen

                            % Check column-wise in that column in d01 to avoid asigning a traj segment start qstart to a traj segment end qend that had already
                            % had a traj segment asigned to it which might be at a shorter distance than the current one.

                            %                                  [d01{:,qstart_chosen}] % chosen column of d01 matrix of distances.

                            % If the found distance in that column is not the minimum one we don't proceed with the asigment:
                            if qend ~= find([d01{:,qstart_chosen}] == min([d01{:,qstart_chosen}]));
                                d01_best{qend,qstart_chosen} = single(rej+0.1); % we re-set that distance to rej+0.1 (in case we need to distinguish these cases from cases rej later on).
                                % if rejected, it moves on to next element in sorted_row (loop ii).

                            else % if qend really corresponds to the minimum distance (best asignment column-wise):

                                d01_best(qend,qstart_chosen) = d01(qend,qstart_chosen); % Pass value on to the "best-asignments" distance cell array d01_best.
                                % d01_best must have only one value different from rej per row and per column. All elements initially have a pre-assigned value of rej.

                                % Re-set values of previous assignments which were not optimum:
                                for q0 = 1:(qend-1) % loop through previously filled-up elements in that chosen column qstart_chosen:
                                    if d01_best{q0,qstart_chosen} < rej % for all elements in that column of d01_best except for the best asignment q0:
                                        d01_best{q0,qstart_chosen} = single(rej+0.2); % re-set larger (not best) distance values in that column.
                                    end
                                end

                            end

                        end
                    end
                end


            else % if qend really corresponds to the minimum distance (best asignment column-wise):

                d01_best(qend,qstart_chosen) = d01(qend,qstart_chosen); % Pass value on to the "best-asignments" distance cell array d01_best.
                % d01_best must have only one value different from rej per row and per column. All elements initially have a pre-assigned value of rej.

                % Re-set values of previous assignments which were not optimum:
                for q0 = 1:(qend-1) % loop through previously filled-up elements in that chosen column qstart_chosen:
                    if d01_best{q0,qstart_chosen} < rej % for all elements in that column of d01_best except for the best asignment q0:
                        d01_best{q0,qstart_chosen} = single(rej+0.2); % re-set larger (not best) distance values in that column.
                    end
                end

            end

        end

    end

    %      d01_best

end


% List of best assignments of linked trajectory segments:
[u v] = find(cell2mat(d01_best)<rej); % find positions of best assignments (elements in d01_best <rej). u is row, v is column.
best_assignmts = sortrows([u v],1); % sort the list of best assignments of trajectory segments by the first column (end of traj segment).

% [u v]


% --------------------------------------------------------
% GROUP BEST ASSIGNMENTS WHICH CORRESPOND TO THE SAME SPOT:

% Eg. Example with 16 trajectories in total for which d01_best only has 6 elements < rej,
% which are saved into best_assignmts: [5 16; 7 11; 8 13; 10 12; 11 15; 12 14].
% At the end of the following loop to group assignments, we will have the cell array best_assignmt_groups:
% {5 16; 7 11 15; 8 13; 10 12 14}, for which trajectory segments linked into full trajectories are: 5-16,
% 7-11-15, 8-13 and 10-12-14.

best_assignmt_groups = num2cell(best_assignmts,2);
% preliminary copy of best_assignments from which we will delete the ones that we have already included into another assignment as a group.
% convert matrix to cell array of as many 1x2 matrices as assignments, in order to be able to modify elements to store matrices of different sizes later on.

jj = 1; %jj is index for accepted best assignments.
while jj < size(best_assignmt_groups,1)

%     jj

    first_column = []; % make list of elements in first column of assignments.
    for kk = 1:size(best_assignmt_groups,1)
        first_column = [first_column; best_assignmt_groups{kk}(1)]; % append first element of other assignemnts.
    end

%     first_column

    row_link = find( first_column == best_assignmt_groups{jj}(end) ); % row corresponding to another found assignment which links to the present one.
    % note that in best_assignmt_groups all competing assignments have been solved already.

%     row_link

    if isempty(row_link) % if there is not another assignment which links to the present one:
        jj = jj+1; % we leave best_assignmt_groups unchanged and advance index jj to move on to next assignment in best_assignmt_groups.
    else % if present assingment jj can be grouped (linked) to another assignment:
        best_assignmt_groups{jj,:} = [best_assignmt_groups{jj,:} best_assignmt_groups{row_link}(2)]; % group the linkable assignments into one and save to assignment jj.
        best_assignmt_groups(row_link) = []; % delete from best_assignmt_groups the row corresponding to assignment grouped to the current one (use () instead of {} to delete!).

        % do not advance jj index, try to find a possible following link
        % for current grouped assignment.
```

```matlab

%          best_assignmt_groups
%          best_assignmt_groups{jj}

    end


end
% the final list of grouped assignments is best_assignmt_groups.
% ----------------------------------------------------------------



% -----------------------------------------------------------
% CREATE FULL TRAJECTORIES AFTER LINKING TRAJECTORY SEGMENTS:

% Create fully linked trajectories:
% Modify Traj Number in separated_trajs data to give same Traj number to trajectories
% which have been grouped and linked as given by best_assignmt_groups.
% Then sort again by Traj Number again.
% In the previous example, after linkin and grouping trajectories, we would
% end up with only 10 trajectory numbers 1 to 10 instead of the previous 16.

% Reminder: the result separated_trajs is a cell array with as many matrices (of
% different sizes) as trajectories. E.g.: separated_trajs{tr} gives the
% data for trajectory "tr". E.g. separated_trajs{tr}(:,1) gives all the
% CentreX values (1st column) of trajectory number "tr".
% Note that the first row in each trajectory is reserved for adding column
% labels latter!!!
% Reminder: traj_column is the number of column containing trajectory numbers.

separated_trajs_linked = separated_trajs; % This will be the final result of linked trajectories (we asign it a preliminary↙
value to start with).
old_new_traj_numbers = []; % Matrix to be filled up. Each row will have two elements: the old traj number and the new traj↙
number we will change the old one into.

for m = 1:size(best_assignmt_groups,1) % loop through groups of linked trajectories.
```

```

    first_traj_no = best_assignmt_groups{m}(1); % Traj. Number for first trajectory in group m. We assign that Traj number to all trajectories in group m.

    for n = 2:length(best_assignmt_groups{m}) % loop through all but first traj in the group:
        next_traj_no = best_assignmt_groups{m}(n); % next Traj. Number in group m.
        old_new_traj_numbers = [old_new_traj_numbers; [next_traj_no first_traj_no]]; % List of traj numbers that need to be changed (first column) and assigned new traj numbers (second column). To be used later.
        % Replace all Traj. Numbers in the other trajectory segments in the
        % group by the Traj. Number (first_traj_no) of the first traj
        % segment in group m:
        for p = 1:length(separated_trajs_linked{next_traj_no}(:,traj_column))
            separated_trajs_linked{next_traj_no}(p,traj_column) = first_traj_no;
        end

    end

end

% old_new_traj_numbers

% ------------------------------
% GENERATE FINAL DATA TO EXPORT:

data_to_export_unsorted_2 = [];
for tr = 1:trajs_max % loop through all former trajectory numbers (all elements in cell array separated_trajs_linked):
    data_to_export_unsorted_2 = [data_to_export_unsorted_2; separated_trajs_linked{tr}];
end

data_to_export_sorted_2 = num2cell(sortrows(data_to_export_unsorted_2,traj_column)); % sort by Traj number and convert to cell to be able to add labels.
% -------------------------------



% --------------------------------------
% PLOT FINAL FULL TRAJECTORY RESULTS:
```

```

full_traj_results = traj_results;
% This will be the same full trajectory result (final resutl after linking and grouping traj segments),
% as a structure in order of image frames, so that we can overlay final
% trajectories on the original image sequence.

% Loop through selected frames:
for k = start_frame:end_frame

    frame = image_data(k).frame_data; % extract frame data which is stored in field 'frame_data'.
    frame = double(frame);

    disp(['frame number: ',num2str(k)]) % print frame number to Command Window.

    imshow(frame,[],'Border','tight','InitialMagnification',150); % show image scaled between its min and max values ([]).
    hold on;
    pause(0.5);

%     k

    % Plot final result of full connected trajectories:
    for q = 1:max_no_spots % loop through found spots on each frame:

        %         q

        % full_traj_results(k,q) is a structure containing the found spot
        % characteristics, including fields 'CentreX', 'CentreY' and
        % 'TrajNumber'.
        tr = traj_results(k,q).TrajNumber; % trajectory number corresponding to found spot q on frame k (in old traj↙
results).

        %         tr

        if isempty(old_new_traj_numbers)==0 % if there are traj numbers to be changed:
            if isempty(tr) == 0 % if that spot is not empty (due to padding of structure array):
                pos_to_change = find(old_new_traj_numbers(:,1)==tr); % check if traj number tr for this spot is contained in↙
the list of traj numbers that need to be changed (old_new_traj_numbers).
```

```

        %      pos_to_change

        if isempty(pos_to_change) == 0 % if the traj number tr for this spot is contained in the list of traj numbers that need to be changed:
          full_traj_results(k,q).TrajNumber = old_new_traj_numbers(pos_to_change,2); % change that traj number.
        end
      end
    end

    if  tr > 0  % only plot trajectory numbers > 0 (TrajNumber = 0 is for loose spots):
      % Plot each trajectory in a different colour:
      plot(full_traj_results(k,q).CentreX,full_traj_results(k,q).CentreY,'o','Color',color_list(full_traj_results(k,q).TrajNumber,:),'MarkerSize',10)
      %      % Construct trajectory data to save (convert structure to matrix):
      %      data_to_export = [data_to_export; cell2mat(struct2cell(traj_results(k,q)))']; % append row with the spot data.
    end
  end
  pause(0.2);
  hold off;

end  % loop through selected frames.
% ----------------------------------



%% EXPORT TRAJECTORY DATA:

% Note: the file gets saved into the same directory where the .sif image is
% (sifImagePath).

% % Export unsorted data ("unsrt"):
% output_filename0 = strcat(data_set_label,'_',image_label,'unsrt0.xls'); % output .xls filename (before sorting data by trajectory number)
% xlswrite(output_filename0,data_to_export_unsorted); % write data to excel file.

% % Export initial sorted trajectory data (before linking trajectory segments):
% data_to_export_sorted = [data_to_export_labels; num2cell(data_to_export_sorted)]; % Add as a first row the fieldnames (labels for each column).
% output_filename = strcat(data_set_label,'_',image_label,'sorted0.xls'); % output .xls filename (before sorting data by trajectory number)
% xlswrite(output_filename,data_to_export_sorted); % write data to excel file.

% -------------------------------------
% Export final sorted full trajectory data (after linking and grouping
% trajectory segments):
output_filename = strcat(data_set_label,'_',image_label,'fullTrajs.xls'); % output .xls filename (before sorting data by trajectory number)
data_to_export_sorted_2 = [data_to_export_labels; data_to_export_sorted_2]; % Add as a first row the fieldnames (labels for each column).
% Export parameters used for function "FindTrajects.m" and
% "linkTrajSegments.m" to two other sheets in excel file:
warning off MATLAB:xlswrite:AddSheet % turn warning off when new sheet added to excel file.
% Params for "FindTrajects.m":
dataForSheet1 = [fieldnames(params_for_FindTrajects) struct2cell(params_for_FindTrajects)];
% Params for "linkTrajSegments.m":
dataForSheet2 = [fieldnames(params) struct2cell(params)];

xlswrite(output_filename,dataForSheet1,'params FindTrajects'); % write data with parameters for "FindTrajects.m" to sheet 'params FindTrajects' in excel file.
xlswrite(output_filename,dataForSheet2,'params linkTrajSegments'); % write data with parameters for "linkTrajSegments.m" to sheet 'params linkTrajSegments' in excel file.
xlswrite(output_filename,data_to_export_sorted_2,'Track results'); % write trajectory data to sheet 'Track results' in excel file.
% Note: the file gets saved into the same directory where the .sif image is
% (sifImagePath).

% ---------------------------------------

% % Export also all (non-empty) spots in the input (traj_results) structure,
% % even if they have not been linked into trajectories, i.e.,
% % export "all_data_to_export":
% all_data_to_export_2 = [data_to_export_labels; num2cell(all_data_to_export)]; % Add as a first row the fieldnames (labels
for each column).
% output_filename = strcat(data_set_label,'_',image_label,'allspots.xls'); % output .xls filename (before sorting data by↙
trajectory number)
% xlswrite(output_filename,all_data_to_export_2); % write data to excel
% file.




```