```
%% ----- Exercise 6 - Field data acquisition and analysis -----
% Version: 2022
% Course: TME 192 Active Safety
%         Chalmers
% Author: Alberto Morando - morando@chalmers.se
%         Christian-Nils Boda - boda@chalmers.se
%         Alexander Rasch - arasch@chalmers.se
%         Pierluigi Olleja - ollejap@chalmers.se
%
% Group number: [Group 5]
        % Group member: [Yahui Wu]
        % Group member: [Tianshuo Xiao]
        % Group member: [Nishanth Suresh]

close all
clear all
clc

%% WARNING
% Before using this script, convert the BAG file into a CSV file with the
% function `bag2csv`. For example `bag2csv('2018-10-24-11-11-25_0.bag')`
% The function will convert all LIDAR data and skip any other data (like
% video data etc.)

%% DATA LOADING
% `importLidarData` Fill a structure containing the recorded data. The structure is
as follow:
% +------------+-------
+-----------------------------------------------------------------------+
% | Timestamp_s | [s]   | n x 1 vector, where n is the number of timestamps (or
frames)       |
% | Angle_rad   | [rad] | 1 x n vector, where n is the number of scanned angles.
|
% |             |       | The range of angles is [-1.658063, 1.658063] radians (-
95, +95 deg) |
% | Distance_m  | [m]   | m x n matrix, where m is the number of timestamps (or
frames),      |
% |             |       | and n is the number the scanned angle.
|
% |             |       | Each value represents the distance at a given time stamp
and angle  |
% +------------+-------
+-----------------------------------------------------------------------+

filename_csv = '2018-10-24-11-11-25_0.csv' % <--- Change this one with your own
file!

data = importLidarData( filename_csv );

% Sampling settings
sample_period_s = median(diff(data.Timestamp_s));

%% SELECT THE EVENT OF INTEREST
% Find the relevant events for your dataset using the fuction playLidar().
% With `playLidar` determine what is the optimal FOV to detect the
% obstacles, and write down the start and stop time index (frame) for each event
into `index_of_events`.
% The matrix `index_of_events` should follow the following template:
%
```

```
% +-------+------+
% | start | stop |
% +-------+------+
% |     1 |  123 |
% |   156 |  236 |
% |   ... |  ... |
% +-------+------+

% Uncomment the line to use `playLidar`
% playLidar('2018-10-24-11-11-25_0.bag')

% Store your event indices in a matrix like the following (example)
index_of_events = [290, 403; 756, 806; 1103, 1229; 1555, 1602; 1948, 2107; 2480,
2575;8190, 8328;];%

% Change the FOV limits to what you found that gave the best results
FOV_limits_deg = 15; % <--- Change this with respect to your data!

FOV_limits_rad =  deg2rad([-FOV_limits_deg/2, FOV_limits_deg/2]);

% Extract the index for the columns within the selected field of view (FOV)
FOV_span_idx = [find(data.Angle_rad >= FOV_limits_rad(1), 1, 'first') :
find(data.Angle_rad <= FOV_limits_rad(end), 1, 'last')];

%% PLOT A SNAPSHOT OF THE LIDAR MEASUREMENT
% Select a _single_ frame from one of your events and plot the LIDAR
% measurement
frameNumber = 300;  % <--- Change this with respect to your data!

figure('name', 'Example of FOV')
hold on

% Plot the LIDAR origin
plot(0, 0, 'r^', 'markersize', 10, 'MarkerFaceColor', 'r');

% Plot the distance measurements in the full field of view
[X, Y] = pol2cart(data.Angle_rad+pi/2, data.Distance_m(frameNumber,:));
plot(X, Y, '-k');

% Extract the data in the chosen FOV
[X_fov,Y_fov] = pol2cart(data.Angle_rad(FOV_span_idx)+pi/2,
data.Distance_m(frameNumber,(FOV_span_idx)));
plot(X_fov, Y_fov, '-k', 'linewidth', 2);

% M the closest point in the field of view
x_closest = X_fov(find(Y_fov == min(Y_fov)));
y_closest = min(Y_fov);
plot(x_closest, y_closest, 'xb', 'linewidth', 1.2, 'markersize', 8);

% Plot the boundaries of the selected field of view
plot([0, min(X_fov)], [0, max(Y_fov)], ':r', 'linewidth', 1.5) % left limit
plot([0,max(X_fov)], [0, max(Y_fov)], ':r', 'linewidth', 1.5) % right limit

% For a better visualization you should limit the axis in the figure
set(gca, 'xlim', [-2, 2], 'ylim', [0, 11])
axis equal

xlabel('Distance [m]');
ylabel('Distance [m]');
```

```
legend('LIDAR', 'Data_{all}', 'Data_{FOV}', 'Closest point', 'Boundaries of ','the
field of view')

%% PLOT DISTANCE OF TARGET ACROSS TIME
figure;
hold on
for i=1:length(index_of_events)
    [X_fov_event, Y_fov_event] = pol2cart(data.Angle_rad(FOV_span_idx)+pi/2,
data.Distance_m(index_of_events(i,1):index_of_events(i,2),(FOV_span_idx)));
    Y_target = min(Y_fov_event,[],2);
    X_target = [];
    for j=1:length(Y_target)
        X_target = [X_target;X_fov_event(find(Y_fov_event(j,:)==Y_target(j,1),1))];
    end
    % X_target = smoothdata(X_target);
    % Y_target = smoothdata(Y_target);

    dis = sqrt(X_target.^2 + Y_target.^2);
    dis = smoothdata(dis);
    time = data.Timestamp_s(index_of_events(i,1):index_of_events(i,2))-
data.Timestamp_s(index_of_events(i,1));
    plot(time,dis,'Linewidth',2)
end
xlabel('Time [s]'); ylabel('Distance [m]')


%% PLOT RELATIVE SPEED TO TARGET ACROSS TIME
figure;
hold on
for i=1:length(index_of_events)
    [X_fov_event, Y_fov_event] = pol2cart(data.Angle_rad(FOV_span_idx)+pi/2,
data.Distance_m(index_of_events(i,1):index_of_events(i,2),(FOV_span_idx)));
    Y_target = min(Y_fov_event,[],2);
    X_target = [];
    for j=1:length(Y_target)
        X_target = [X_target;X_fov_event(find(Y_fov_event(j,:)==Y_target(j,1),1))];
    end
    % X_target = smoothdata(X_target);
    % Y_target = smoothdata(Y_target);

    dis = sqrt(X_target.^2 + Y_target.^2);
    time = data.Timestamp_s(index_of_events(i,1):index_of_events(i,2))-
data.Timestamp_s(index_of_events(i,1));

    velocity = diff(dis) ./ 0.05;
    velocity(velocity > 0) = NaN;
    velocity = smooth(velocity);
    velocity(velocity > 0) = NaN;
    plot(time(1:end-1),velocity,'Linewidth',2)
end

xlabel('Time [s]'); ylabel('Relative speed [m/s]')
ylim([-4 2])

%% PLOT TIME TO COLLISION TO TARGET ACROSS TIME
figure;
hold on
ttcTable = [];
```

```
for i=1:length(index_of_events)
    [X_fov_event, Y_fov_event] = pol2cart(data.Angle_rad(FOV_span_idx)+pi/2,
data.Distance_m(index_of_events(i,1):index_of_events(i,2),(FOV_span_idx)));
    Y_target = min(Y_fov_event,[],2);
    X_target = [];
    for j=1:length(Y_target)
        X_target = [X_target;X_fov_event(find(Y_fov_event(j,:)==Y_target(j,1),1))];
    end

    dis = sqrt(X_target.^2 + Y_target.^2);

    time = data.Timestamp_s(index_of_events(i,1):index_of_events(i,2))-
data.Timestamp_s(index_of_events(i,1));
    velocity = diff(dis)./0.05;
    % dis =smoothdata(dis);
    % velocity = smoothdata(velocity);
    velocity(velocity > 0) = NaN;
    velocity = smooth(velocity);
    velocity(velocity > 0) = NaN;
    ttc = dis(1:end-1) ./ -velocity;
    ttc = smooth(ttc);

    ttcTable = [ttc;ttcTable];
    plot(time(1:end-1),ttc,'Linewidth',2)
end

xlabel('Time [s]'); ylabel('TTC [s]')

ylim([0 25]);
xlim([0 9]);
%% --- Question ---
% What safety measure would you use to design a warning that alert the user
% that is about to collide with an obstacle? You  may want to use one of the safety
measure
% you computed in this script of find a more effective one.
% What value of such measure would you use to trigger a warning?
% (Write your answer and enclose it in a comment)
figure(2)
ttc_1 = ttcTable;
histogram(ttc_1);
per = prctile(ttc_1,[5,95]);
xlabel('TTC [s]');
ylabel('Frequency');
% Based on the calculated TTC values and we use this safety measure to design a
warning
% system.Then,we caculate the percentiles and it provide a statistical measure of
the
% distribution of TTC values and can help us set thresholds for triggering
warnings.
% The 5th percentile is 2.23s, the 95th percentile is 60.04s.In designing the TTC
that triggers the warning,
% we considered the situation of most people, so we choose 2.23s as our TTC.
% Considering the system response time(0.1s) and the driver reaction time(1.0s),
% we set the threshold as 3.33s to trigger a warning.
```