```
function path = lab4( port_num, feedback_hz, display_on,kp_passed, kd_passed, v_passed)
%% LAB4 is a skeleton file for feedback control of the iRobot Create 2
%
%   LAB4( port_num, feedback_hz )
%
%   ARGUMENTS:
%   'port_num' - the COM port number for the iRobot Create2 interface. You
%       can get this from device manager.
%   'feedback_hz' - the desired feedback control loop rate. Between 5-10
%       is a good value for this lab. Higher than 20Hz will not do anything
%       productive as this is the limit of the Create's acuators
%   'display_on' - if true, it will display the LIDAR scan. Once you get
%       things running, set to false so it does not slow you down.
%
%   RETURNS:
%   'path' - kx2 vector which is the x-y positions of the robot from each
%       scan.
%
%   This is a skeleton file for the LIDAR lab. It assumes you have already
%   called rosinit(URI) in the workspace, i.e.,
%
%   >> rosinit('http://192.168.0.123:11311');
%
%   SAMPLE USAGE:
%   Example setting the COM port to COM3, the update rate to 10 Hz, and the
%   display of the LIDAR scan to off.
%
%   >> lab4( 3, 10, false );
%
%
% See also iRobotCreate, ROSINIT
%
% Copyright (C) 2016, 2019 by JRS @ Lehigh University

%% Global variables
% These implement the functionality of C's kbhit.
global kbhit;
kbhit = false;

%% Local Variables
% UNCOMMENT: Instantiate a robot
r2d2 = iRobotCreate('Port', port_num, 'Version', 2, 'Hz', feedback_hz);
% Subscribes to the ROS /scan topic
laser = rossubscriber('/scan');
% Parameters for the LIDAR
alpha = (-45:0.5:45)'*pi/180;
% Empty array for storing the path. Faster to pre-allo, but we'll be OK...
path = zeros(0,2);
% Loop counter
i = 0;
```

```
%% Figure setup
close all;
figure('Name','SICK LMS291 LIDAR Stream','KeyPressFcn',@mykbhit);
colormap('jet');

out_timer = tic;
%% Feedback Control Loop
while ~kbhit
    % Timer Start. For maintaining the feedback Hz rate.
    in_timer = tic;

    % Get a LIDAR scan.
    scan_data = receive(laser,10);
    rho = scan_data.Ranges;
    gamma = scan_data.Intensities >250;

    %used to get alpha and rho values for the origin and end point
    %my_alpha = mean(alpha(gamma));
    %my_rho= mean(rho(gamma));

    my_alpha = alpha(gamma);
    my_rho = rho(gamma);

    %origin_alpha =0.5105;
    %origin_rho = 4.0150;
    %end_alpha =-0.3971;
    %end_rho= 4.3250;

    origin_x = 3.5031;
    origin_y = 1.9618;
    %end_x = 3.9885;
    %end_y = -1.6725

    %used to calculate the angle of frame wrt to lidar
    %frame_theta = atan2(end_y - origin_y, end_x - origin_x);
    frame_theta =  -1.4380;

    %rotation matrix
    A = [cos(frame_theta) -sin(frame_theta) origin_x; sin(frame_theta) cos(frame_theta)↙
origin_y; 0 0 1];
    invA = inv(A);

    %convert alpha and rho values to x and y
    [x, y] = pol2cart(my_alpha, my_rho); % X and Y translation of our frame with↙
Lidar's frame

    %obtaining the cluster and resolving it into two points
    [~,kmeans_result] = kmeans([x y], 2);
    p1 = [kmeans_result(1); kmeans_result(3); 1];
```

```
    p2 = [kmeans_result(2); kmeans_result(4); 1];

    %converting the points wrt to our frame
    p1 = invA * p1;
    p2 = invA * p2;

    mid_x = mean([p1(1) p2(1)]);
    mid_y = mean([p1(2) p2(2)]);

    %obtaining the theta
    theta = atan2(p2(2) - p1(2), p2(1) - p1(1));

    %velocity to be passed to the robot
    v = v_passed;
    %calculating the omega to pass on to the robot
    kp = kp_passed;
    kd = kd_passed;
    w = (-1*kd*tan(theta))-((kp*mid_y)/(v*cos(theta)));

    %constraint on  w
    if theta > 1.05 && w >0
        w = 0;
    elseif theta < -1.05 && w<0
        w = 0;
    end

    % Display the points only when you are debugging
    if display_on
        if exist('h','var')
            delete(h);
        end
        disp(x);
        disp(y);
        disp(gamma);
        h = scatter(x,y,12,gamma,'filled');
        axis([0 5 -5 5]);
        drawnow;
     else
%           disp("w");
%           disp(w);
%           disp("theta");
%           disp(theta);
%           disp("X");
%           disp(x);
%           disp("Y");
%           disp(y);
%
%           disp("alpha");
%           disp(my_alpha);
%           disp("rho");
```

```
%          disp(my_rho);
%          disp(frame_theta);
%          disp(p1);
%          disp(p2);
%          disp(theta);
    end

    %% Add your code here to get the robot pose and run the controller

    % UNCOMMENT: Update the path variable with the new robot location estimate
    path = [path; [toc(out_timer) mid_y]];

    % UNCOMMENT: Your controller code goes here.
    r2d2.setvel(v,w);

    % Timer Stop. Pause for any time remaining on this cycle.
    i = i+1;
    dt = toc(in_timer);
    pause( 1/feedback_hz - dt );

    %Once the robot has passed the end point, break;
    if mid_x > 3.5
        v = 0;
        w=0;
        r2d2.setvel(v,w);
        disp(mid_x);
        disp(mid_y);
        break;
    end
    %UNCOMMENT: Check if we've hit anything, and if yes stop and exit.
     if r2d2.isbumped()
         r2d2.setvel(0,0);
         break;
     end

end
%% Actual Feedback Loop Update Rate
% This is to get the effective scan rate so we can see if we have any
% bottlenecks
dt_out = toc(out_timer);
fprintf('Actual scan rate was %0.2f\n',i/dt_out);

%% Output figure. Edit as necessary
figure;
hold on;

% This is our blue tape line
%plot(min(path(:,1)),min(path(:,1)),'b-','linewidth',2);

% This is the actual path
```

```
columnVal = path(:,1);
columnVal2 = path(:,2);
plot(columnVal,columnVal2,'ro');

% Set the axis and label properties
axis tight;
xlim([0 18]);
ylim([-0.45 0.45]);
xlabel('time (s)');
ylabel('y (m)');



function mykbhit( ~, ~ )
%% MYKBHIT simulates kbhit in 'C'
global kbhit;
kbhit = true;
```