Table of Contents

MATLAB 2D Rover Simulation - Pure Pursuit Control 1
Define Environment and Path 1
Simulate Pure Pursuit Control 1
Function Definitions 1

MATLAB 2D Rover Simulation - Pure Pursuit Control

Simulates a rover following a predefined path using Pure Pursuit Control.

```
clc; clear; close all;
```

```
Error using evalin
Undefined function 'PPC' for input arguments of type 'char'.
```

Define Environment and Path

```
T = 0.1;          % Time step (s)
steps = 200;      % Simulation steps
L = 1.5;          % Rover wheelbase (m)
v = 1.0;          % Constant velocity (m/s)

% Define predefined path (sinusoidal curve)
path = [linspace(0, 20, 50)', sin(linspace(0, 2*pi, 50))' * 2 + 5];

% Define obstacles (static points)
obstacles = [5, 6; 10, 4; 15, 7];
```

Simulate Pure Pursuit Control

```
fprintf("Running Pure Pursuit Simulation...\n");
pure_pursuit_simulation(path, L, v, T, steps, obstacles);
```

Function Definitions

```
function pure_pursuit_simulation(path, L, v, T, steps, obstacles)
    pos = path(1, :);       % Initial position at the start of the path
    theta = 0;              % Initial heading angle (radians)
    lookahead_dist = 2.0;   % Lookahead distance

    figure; hold on; grid on;
    h1 = plot(path(:,1), path(:,2), 'b--', 'LineWidth', 1.5); % Path
    scatter(obstacles(:,1), obstacles(:,2), 100, 'k', 'filled'); % Obstacles
    h2 = scatter(pos(1), pos(2), 'ro'); % Initial rover dot
    xlabel('X'); ylabel('Y');
    title('Pure Pursuit Control');
```

```
    legend([h1, h2], {'Reference Path', 'Rover'}); % Only one red dot in
legend

    for i = 1:steps
        % Find lookahead point
        target_idx = find_lookahead_point(pos, path, lookahead_dist);
        if isempty(target_idx)
            target_idx = size(path, 1); % Use last point if none found
        end

        target_point = path(target_idx, :);

        % Calculate steering angle
        alpha = atan2(target_point(2) - pos(2), target_point(1) - pos(1)) -
theta;
        delta = atan2(2 * L * sin(alpha) / lookahead_dist, 1);

        % Update rover state using bicycle model
        theta = theta + (v / L) * tan(delta) * T;
        pos = pos + v * T * [cos(theta), sin(theta)];

        % Plot current position and orientation (invisible to legend)
        scatter(pos(1), pos(2), 'ro', 'HandleVisibility', 'off');
        quiver(pos(1), pos(2), 0.5*cos(theta), 0.5*sin(theta), 'r',
'LineWidth', 1.5, 'HandleVisibility', 'off');
        drawnow;
        pause(0.05);

        % Check if reached end of path
        if norm(pos - path(end, :)) < 0.1
            break;
        end
    end
end

function target_idx = find_lookahead_point(pos, path, lookahead_dist)
    % Calculate distances from current position to all path points
    distances = sqrt((path(:,1) - pos(1)).^2 + (path(:,2) - pos(2)).^2);

    % Find the closest path point
    [~, closest_idx] = min(distances);

    % Search ahead from the closest point
    target_idx = [];
    for i = closest_idx:length(distances)
        if distances(i) >= lookahead_dist
            target_idx = i;
            break;
        end
    end

    % If no point found, use the last point
    if isempty(target_idx)
        target_idx = length(distances);
```

```
    end
end
```

Published with MATLAB® R2024b