Pure pursuit overview

In class, we showed that the pure pursuit algorithm for driving along a line can be straightforward to implement if we have access to the transformation $T^L_R$ from the robot frame to the line frame. We assume the line frame is defined such that the line lies on the $x$-axis of the coordinate frame.

We also assume the robot-to-line transformation is given by

$$T^L_R = \begin{bmatrix} 
    c\theta & -s\theta & x \\
    s\theta & c\theta & y \\
    0 & 0 & 1 
\end{bmatrix}$$

So the point $(x, y)$ specifies the position of the robot in the line frame.

To run the pure pursuit controller, then:

1. Compute the point $p_c$, the closest point on the line to the robot. In the line frame, the coordinates are simply $(x, 0)$.

2. March ahead of $p_c$ along the line by some distance $\alpha$ to obtain $p_d$, the pursuit point for the robot. The line-frame coordinates of $p_d$ are $(x + \alpha, 0)$.

3. Issue the controls

$$\dot{x}_R = k_x, \quad \dot{\theta} = k_\theta \frac{c_y}{c_x}$$

where the point $(c_x, c_y)$ are the coordinates of $p_d$, expressed in the robot frame – that is, the coordinates obtained by mapping the point $(x + \alpha, 0)$ through the inverse of the transformation $T^L_R$. 
1. Simulated pure pursuit

Write a simulator to evaluate some control strategies for pure pursuit. Assume you have a differential drive robot whose body-frame velocities $\dot{x}_R$ and $\dot{\theta}_R$ can be commanded directly (note: you can probably re-use some code from Homework 1, and throw away the parts that deal with $v_L$ and $v_R$).

a. Start the robot at $(x, y, \theta) = (0, -0.5, 0)$. Set the gains to $\alpha = 0.2$, $k_x = 0.1$ and $k_\theta = 2.0$, and simulate for 30 seconds using Euler’s method with $\Delta t = 0.01$ second. Graph the motion of the robot in the world frame on an $(x, y)$ plot. Make sure your plot has equal scaling on the $x$ and $y$ axes (in MATLAB, for instance, use the `axis equal` command).

b. Simulate what happens with a lookahead distance too small ($\alpha = 0.05$) and too large ($\alpha = 1.0$). Submit plots for each.

c. Restore $\alpha = 0.2$ and simulate what happens when you clamp the angular velocity to be in the $\pm 0.15$ rad/s range (i.e. enforce a maximum rotational velocity limit). You should see some oscillation in the robot’s motion. Modify the $\alpha$ and/or $k_\theta$ gains to fix the behavior, and submit plots of both the overshooting and fixed behavior, with the second plot labeled with the new gains.

Please hand in printouts of all five plots, along with whatever code you used to generate them.