Instruction for Project 1

Lookahead Pure Pursuit Control

1. Introduction

Our final project aims to develop a high-performance race car capable of navigating a designated race track quickly and robustly. To achieve this, our upcoming project involves constructing a controller for the race car, taking RIDAR input and generating appropriate actions. To facilitate this training process, we require expert demonstrations from an agent with proficient track navigation skills. These demonstrations will serve as valuable training data for our future algorithm.
In this project, we will design a race car that can efficiently navigate a predefined race track. Waypoints are strategically positioned along the track, and the race car will strive to reach each subgoal using a pure pursuit algorithm. To execute this algorithm, we will employ a PID controller to guide the race car towards the intended goal waypoint at each time step.

2. PID control

2.1. Main Theory

A PID controller (proportional–integral–derivative controller) is a control loop feedback mechanism widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an error value e(t) as the difference between a desired setpoint and a measured process variable and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively) which give their name to the controller. The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining u(t) as the controller output, the final form of the PID algorithm is

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau)\,\mathrm{d}\tau + K_\mathrm{d} \frac{\mathrm{d}e(t)}{\mathrm{d}t},$$

where $e(t)$ is the error between the setpoint and the current possess variable and K_p, K_i, K_d are the proportional gain, the integral gain and the derivative gain respectively.

The proportional term produces an output value that is proportional to the current error value. The proportional term mainly tries to adjust the error term e(t) to zero. The proportional response can be adjusted by multiplying the error by a constant Kp, called the proportional gain constant. A high

proportional gain results in a large change in the output for a given change in the error, but the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain Ki and added to the controller output. The integral term accelerates the movement of the process towards setpoint and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the setpoint value. The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain Kd. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, Kd. Derivative action predicts system behavior and thus improves settling time and stability of the system.

2.2. Discretization

In the PID controller, the integral term can be approximately discretized, with a sampling period Δt, as

$$\int_0^{t_k} e(\tau)d\tau = \sum_{i=1}^{k} e(t_i)\Delta t$$

and the derivative term can be discretized as

$$\frac{de(t_k)}{dt} = \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

Therefore, the control value u(t) can be approximated as

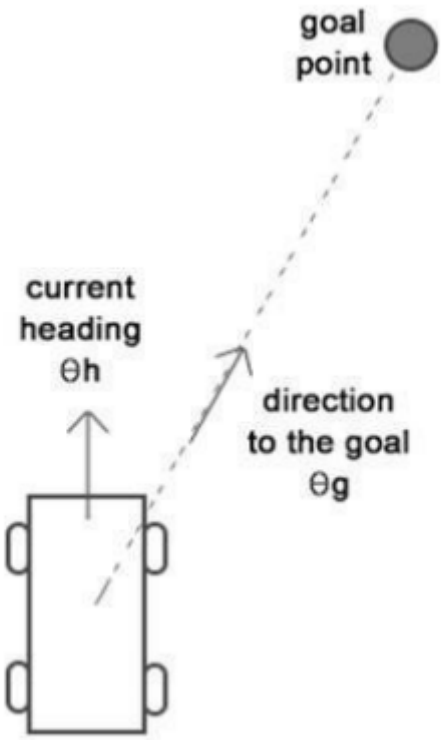
$$u(t) = K_p e(t) + K_i(t)\Delta t \sum_{k}^{t-1} e(k) + \frac{K_d}{\Delta t}(e(t) - e(t-1))$$

or in recursion relation form.

$$u(t) = u(t-1) + (K_p + \frac{K_d}{\Delta t})e(t) + (-K_p + K_i\Delta t - 2\frac{K_d}{\Delta t})e(t-1) + \frac{K_d}{\Delta t}e(t-2)$$

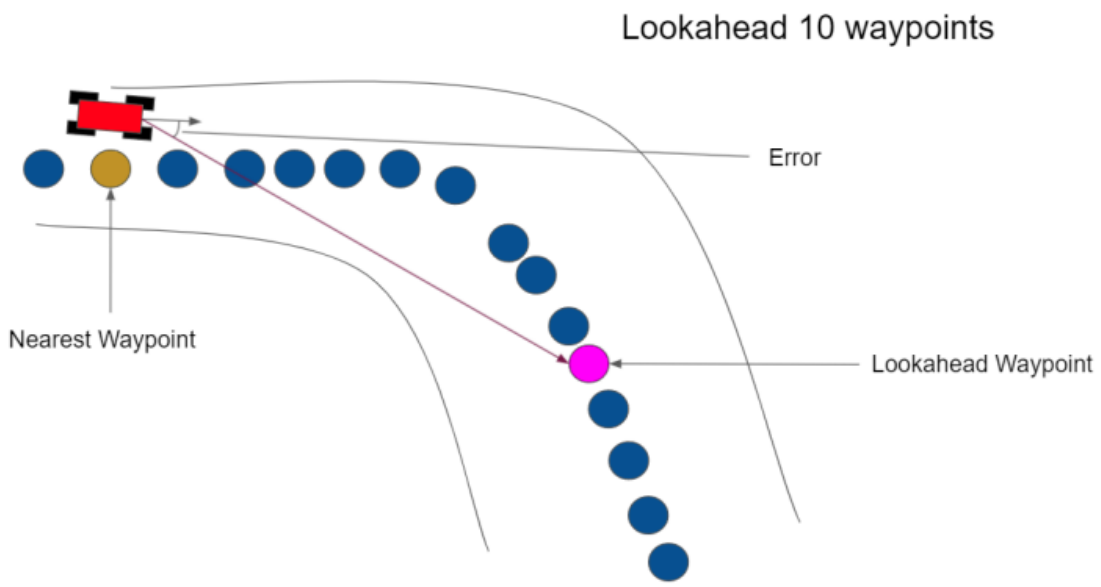
You should control the heading direction of the car to make it follow a given path. So, in this case,

the variable to process is the heading direction of the car. To make the car reach the goal point, the car heading should face the goal point. You can calculate the direction to the goal point for your car in polar coordinate. This value becomes the setpoint which the controller should track. To summarize, the process value which to control is θ_h, and the set point can be θ_g, then $e(t) = \theta_g(t) - \theta_h(t)$.



3. Lookahead Pure pursuit

In this project, students are asked to use lookahead pure pursuit. In the original pure pursuit, the race car tries to reach a fixed waypoint, and when reached, the subgoal is changed to the next waypoint. In this project, this is modified a bit. In this project, the race car finds the nearest waypoint from its current position. Then, the race car tries to go to the 10th next waypoint from its nearest waypoint. Students will be asked to code towards this lookahead waypoint using a PID controller.



4. Assignment

RLLAB_project1.py contains the pure pursuit code needed for this project. You should complete the given python code with TODOs and finetune the parameters for lookahead prepursuit.

1) Change the TEAM_NAME on the top of the code to your team's name. Otherwise, we can't grade your code.
2) Find the nearest waypoint, calculate the error between the race-car heading and the direction vector between the lookahead waypoint and the race-car.
3) Determine the input steering using this error through the PID controller.
4) Use Kp, Ki, Kd to tune your PID controller. You can change other hyperparameters if you want to.
5) Calculate input velocity of the race-car appropriately in terms of input steering.

5. Submission

Please follow the instructions below to download the project files and submit them. From project 1, we won't use ETL for submission and your project score is determined by the result in our project web server.

1) Make a github repository on https://github.com/ and invite our TA(ID: e-j-one)
2) Visit our project github repository(https://github.com/rllab-snu/Intelligent-Systems-2023) and follow the instructions in README.md

3) Make your local git repository in … /Intelligent-Systems–2023/sim2real/project/IS_{your team name}/
4) After uploading, visit our project web server(http://147.46.116.112:36507/) and enter the query as instructed in the server usage tutorial.
5) You can check the leaderboard and the result of your query in the web server.
6) Good luck ~

Don't forget to invite your TA to your team repository and upload the completed code there before entering the query.