Homework 5

ME 416 - Prof. Tron

2024-04-18

The goal of this homework is implement a simple PID controller to make your robot follow the line while moving forward. This homework combines several pieces from previous homework and in-class activities.

Problem 1: PID controller node for line following

Preparation. We will use the functions that we prepared throughout the semester to build a node that aims to command the speed of the robot to make sure that the line is always at the center of the image.

> File name: `controller_line.py`
> Subscribes to topics: `/image/centroid` (type `geometry_msgs/PointStamped`)
> Publishes to topics: `robot_twist` (type `geometry_msgs/Twist`), `control_error` (type `std_msgs/Float64`)
> Description: See the questions below for a detailed description of this node.

Import `robot_model` and `controller` from previous homework assignments and in-class activities.

Question report 1.1. In the `__init__`( ) method for the class for the node, perform the following initializations:

1) Initialize the attributes `lin_speed`, `gain_kp`, `gain_kd`, and `gain_ki`; for now set all of them to zero, they will be tuned later.
2) Initialize the subscriber and two publishers.
3) Initialize an object `pid` from the class `controller.PID` using `gain_kp`, `gain_kd`, and `gain_ki` as inputs.
4) Initialize an object `stamped_msg_register` from the class `robot_model.StampedMsgRegister`.

Include a listing of the relevant code in your report, highlighting or commenting on the location of each item from the list above.

Question report 1.2. Write a callback for the subscriber that performs the following actions:

1) Compute the variable `error_signal` as the difference between the value of the field `point.x` in the received message, and half of the width of the camera image (the image size is $320 \times 240\,\mathrm{px}$).

2) Publish the value of `error_signal` to the topic `/control_error`.
3) Use `stamped_msg_register` to obtain `time_delay`, the time difference between the stamps of the current `PointStamped` message received by this callback, and the previous one (this is similar to what was done in a previous in-class activitya and for the odometry computation in Homework 4).
4) Initialize a message `msg` of type `Twist`.
5) Set `msg.linear.x` to `lin_speed`.
6) Set `msg.angular.z` to the sum of the calls to `pid.proportional`( ), `pid.derivative`( ), and `pid.integral`( ).
7) Publish `msg`.

Include a listing of the relevant code in your report, highlighting or commenting on the location of each item from the list above.

Question optional 1.1. Write a launch file `controller_line_launch.py` that launches the following nodes:

- `motor_command.py` from Homework 2.
- `image_segment.py` from Homework 5.
- `controller_line.py` from this homework.

Question optional 1.2. Possible tricks to improve the processing speed:

- Use the function `np.median`( ) to compute the median instead of sorting the array.
- Modify the functions in `image_processing.py` so that the segmentation and the centroid computation are performed only on the bottom part of the image.
- In `image_centroid_horizontal`( ), index the pixels in the image as `img[idx_row,idx_col]` instead of `img[idx_row][idx_col]`. Similarly, for the outer loop do not use something along the lines of `for row in img:`. In both cases, this will avoid the situation where Python creates intermediate data structures or makes intermediate copies for processing the image in pieces.
- Even better than the point above, look up the function `np.where`( ).
- Resize the image to a lower resolution.

Problem 2: PID tuning

Goal The following questions are designed to tune the gains of the controller. Eventually, our controller is designed to move forward at a constant speed (`lin_speed`), and adjust the angular velocity to keep the track in the middle of the image. The proportional gain is the main gain that drives the error to zero, and will be tuned first. We then add the derivative and integral contributions to improve performance.

Question video 2.1. Since tuning the proportional gain while the robot is moving along the track is cumbersome, we will have the robot turning in place. For this, ensure that `lin_speed`, `gain_kd` and `gain_ki` are set to zero, and initially set `gain_kp=1e-3`. Set the robot on a line in the track.

Run the files `motor_command`, `image_segment`, and `controller_line` (or launch `controller_line_launch.py`). Run `rqt` to show the topic `/image/segmented`. Adjust the value of `gain_kp` (*restarting the node after each change*) so that the robot rotates around itself until the line gets centered in the image frame as fast as possible but with minimal overshoot/oscillation. When you are satisfied, take a video of the robot, and a screen capture of the `/image/segmented` showing that the controller is effective.

Question optional 2.1. Use `ros2 topic echo` to visualize the messages on the topic `/controller_error` (with the tracking error) and `/robot_twist` (with the computed control). While keeping the robot in your hand, check that the sign and behavior of the two make sense (e.g., when the line is in the left half of the image, the angular velocity is positive, i.e., the robot tries to turn left).

Question video 2.2. Repeat the previous question, but this time start with `lin_speed=0.5`, and adjust both `lin_speed` and `gain_kp` such that the robot can follow the track as fast as possible. You should notice that when `lin_speed` is too high, the line goes outside the field of view before the controller can catch up with the turn.

Slowly increase first `gain_kd`, and then also `gain_ki` (suggested increments are in the order of `1e-4` to `1e-3`). You should notice that this allows the robot to better follow the curves, so that you can in turn increase `lin_speed`, and possibly reduce `gain_kp`. Keep tuning all the gains until you are satisfied with the result. Take a video of your best results, and include in the report the final values you settled upon. Ideally, aim for the robot to perform at least one lap of the track.

Hint for question report 1.2: The structure of this node is similar to the nodes from Homework 1.

Hint for question report 1.2: You can look at the node `signal_generator` from the last in-class activity to check how to publish to topics with type `Float64`.

Hint for question report 1.2: Remember that the first time you use the object `stamped_msg_register`, `time_delay` will be `None`, so you cannot use it in the computation of the derivative and integral terms.

Hint for question video 2.1: You might need to flip the sign of the control in `msg.angular.z`, depending on how you compute the error. Make also sure that you have the correct value for the image width.