Control and Trajectory Tracking for Autonomous Vehicles

REVIEW | CODE REVIEW | HISTORY

Meets Specifications

Excellent submission, congratulations Jonathan on passing the project by meeting all the specifications very well. 🎉🎉

Good job in experimenting and coming up with working solution from your end. I truly believe that tuning PID parameters is the most challenging part of this project. 👍

I would especially like to be given advice on parameter tuning for the PID controller (both steering and throttle commands). I would also like to be given information on how these individual gain values relate to the intended actuations (i.e., steering / throttle) and how — when selected carefully, they influence the desired steering / throttle behaviour optimally.

Please go through this interesting paper which should help in addressing your query above.

Please also advise me on auto-tuning strategies. I would also like to have any constructive feedback on my current choice of parameters based on my experiment run results / error plots. I am slightly familiar with strategies that exist in literature (e.g., Ziegler-Nichols, Relay Feedback by Åström, Cohen-Coon method), but am not sure if these may be used here. I have also learned to use transfer functions to address controller stability via poles-zeros; if this is a valid technique that can be used in this project, I would kindly ask for your help in discovering the related transfer function and stability literature related to this PID controller for the self-driving car.

Your current choice of PID parameters and tuning is great and a perfect trajectory is not expected here. The coefficients are working stable to drive the ego car through all obstacles successfully and reach the end of road. Please go through this video on how gains behave and impact vehicle performance.
There is an interesting discussion here over knowledge centre which would help to address your queries related to transfer function.

Finally, please let me know if it is possible to implement Twiddle with respect to the codebase of this project. I have only come across discussion of this idea, but have not yet seen any effort been made to successfully integrate this approach.
It is possible to implement Twiddle with respect to current project however it is not in scope of the current implementation.

You have met all the specifications, but don't stop here, keep experimenting. Additionally, you may take a look at the below video on Motion Planning in a Complex World - MIT Self-Driving Cars here. Trajectory tracking is also one research topic, an interesting research paper on same can be read here.

Code quality

✓ The project requires you to run commands on the terminal and to compile and test code through command lines on a Linux terminal.

Good work ensuring that your code compiles successfully and there are no errors in terminal when running the solution.

```
/home/workspace/nd013-c6-control-starter/project/pid_controller/main.cpp:422:90: warning: unused parameter 'length' [-Wu
used-parameter]
  h.onDisconnection([&h](uWS::WebSocket<uWS::SERVER> ws, int code, char *message, size_t length)
                                                                                         ^~~~~~
[ 16%] Building CXX object CMakeFiles/pid_controller.dir/behavior_planner_FSM.cpp.o
[ 25%] Building CXX object CMakeFiles/pid_controller.dir/motion_planner.cpp.o
[ 33%] Building CXX object CMakeFiles/pid_controller.dir/cost_functions.cpp.o
[ 41%] Building CXX object CMakeFiles/pid_controller.dir/pid_controller.cpp.o
[ 50%] Linking CXX executable pid_controller
[100%] Built target pid_controller
(venv) root@25f0576a5602:/home/workspace/nd013-c6-control-starter/project/pid_controller#
```

Reference on cmake

cmake is a generator of buildsystems while make is a buildsystem. For more information on them, on these two you might want to check out resources.

- Cmake FAQS : This is a useful page to understand cmake frequently asked questions and answers.
- Youtube set of tutorials on using make and writing Makefile - a video to understand cmake usage
- Using make and writing Makefiles - official website of cmake organisation.

✓ The README file includes a summary of the project, how to run the scripts, and an explanation of the files in the repository. Comments are used effectively.

Good job in including the required answers document, you have shared relevant graphs as well.

One way to improve your write up will be to actually discuss and contrast the actual behaviour of the car with the expected behaviour of the car under various conditions including a P controller, a D controller and other types of controllers.
You should also refer to the following links that have some more information on the role of each component.

Reference links

- Controlling Self Driving Cars
- PID controller
- How do the PID parameters (Kp, Ki, and Kd) affect the heading of a differential driving robot when they are increased individually?

Build the PID controller object

✓ The scripts "main.cpp", "pid_controller.cpp" and "pid_controller.h" compile without errors and when "run_main_pid.sh" runs without error.

- The scripts "main.cpp", "pid_controller.cpp" and "pid_controller.h" compile without errors ✅
- "run_main_pid.sh" runs without error ✅

✓ The function "UpdateError" should compute the derivative and integration part of the PID. The function updates the errors for the proportionate, integral and derivative terms.

Nice work, update error computation has been correctly done. ✅
You have computed a discretized discretized integral (additive sum) with the use of the delta_time variable, you could also discretize derivate part.

✓
- The file "pid_controller.h" should be completed as needed and the variable created should be correctly used in the "pid_controller.cpp" file.
- The "pid_controller.cpp" methods should all be completed.
- The PID object instances are well initialized in the "main.cpp" code.

✅ The "pid_controller.h" file contains the declaration of the variables used in the "pid_controller.cpp".
✅ The objects instances are created in main.cpp with the initial values for the constants Kp, Ki and Kd initialized.
✅ The code compile successfully and necessary methods are implemented in pid_controller.cpp.
✅ Necessary checks needed in UpdateError or TotalError are being done.

Adapt the PID controller to the simulation

✓ The script uses the variables described in the README and computes the error between the position and the trajectory for the speed and the yaw.
The code is clearly commented.

✅ The way the error is computed should match the code solution.
✅ The code should contain comments.

✓ The error is sent to the PID correctly and the line:
"// modify the following line for step 3
error_steer = 0;" is commented.
Multiple values of PID parameters are tested and the best is chosen.

✅ The code is able to compile and run successfully.
✅ The code is able to perform successfully as car navigates its way well.
✅ Multiple values of PID parameters are tested and the best is chosen.

Evaluate and analyze the PID controller

✓ The plots are provided in the report.

✅ Plots are provided and show that the simulation worked (numbers were recorded)
✅ Sufficient data points are collected and car drives well.

✓ A few lines explaining the plots and describing what is plotted: the variables, the phenomenon shown.

✅ Explanation of what is represented in the plot.
✅ Analysis of what is changing when parameters changed.

✓ A few lines answer the second question: What is the effect of the PID according to the plots, how each part of the PID affects the control command?
The answer explains the role of the different parts of the PID.

✓ The questions are answered with justified explanations:

- How would you design a way to automatically tune the PID parameters?
  This is an open question, the coherence and justification of the answer is valued.
- PID controller is a model free controller, i.e. it does not use a model of the car. Could you explain the pros and cons of this type of controller?
  Find at least 2 pros and cons for model free versus model based.
- (Optional) What would you do to improve the PID controller?
  This is an open question, the coherence and justification of the answer is valued.

The questions are answered with justified explanations:

✅ How would you design a way to automatically tune the PID parameters?
The answer is coherent, clear and well explained.

✅ PID controller is a model free controller, i.e. it does not use a model of the car.

Here are some examples

Pros: no need to understand the system, adapted to a complex system, computationally more efficient
Cons: difficult interpretation of the controller's behaviour, cannot predict how the system will react to an unknown situation. Tuning or learning the controller using data can be slow. Uncertainty of the controller (no guarantees)

DOWNLOAD PROJECT

RETURN TO PATH