CarND-Controls-PID

Self-Driving Car Engineer Nanodegree Program

PID effections

Let's look deeper on the PID parameters.

P part

Control only with p based controller , the reaction of car was pretty interesting. When the car ran with small cte, it can handled the little cte with small oscillation. But when the car turned the curve, the cte became larger and the controller began to control with big range of oscillation.

I part

I think the only I controller's reaction time is also too long. Reaction time can be handled by adding proportional control. It sum a lot of cte and generated big control value and then repeated the previous step again.

D part

The D based controller's with small cte change rate, generated little control value.

PID with high velocity

- throttle 90% appropriate PID coefficient was tuned. Then applied the value to controller with throlle of 0.9 (90%). Controller generate pretty good steer value in straight road. But once in the curve, controller started to generate steer value that made vehicle oscillation.

PID with low velocity

- throttle 30% Tuned the parameters(kp,ki,kd) appropriately and apply to the code.The output was pretty good. The controller generate meaningful steering value to vehicle. So it can drive the car ran whole track.

Dependencies

- cmake >= 3.5
- All OSes: click here for installation instructions

- make >= 4.1(mac, linux), 3.81(Windows)
  - Linux: make is installed by default on most Linux distros
  - Mac: install Xcode command line tools to get make
  - Windows: Click here for installation instructions
- gcc/g++ >= 5.4
  - Linux: gcc / g++ is installed by default on most Linux distros
  - Mac: same deal as make - [install Xcode command line tools]((https://developer.apple.com/xcode/features/)
  - Windows: recommend using MinGW
- uWebSockets
  - Run either `./install-mac.sh` or `./install-ubuntu.sh`.
  - If you install from source, checkout to commit `e94b6e1`, i.e.

    ```
    git clone https://github.com/uWebSockets/uWebSockets
    cd uWebSockets
    git checkout e94b6e1
    ```

    Some function signatures have changed in v0.14.x. See this PR for more details.
- Simulator. You can download these from the project intro page in the classroom.

Fellow students have put together a guide to Windows set-up for the project here if the environment you have set up for the Sensor Fusion projects does not work for this project. There's also an experimental patch for windows in this PR.

Basic Build Instructions

1. Clone this repo.
2. Make a build directory: `mkdir build && cd build`
3. Compile: `cmake .. && make`
4. Run it: `./pid`.

Tips for setting up your environment can be found here

Editor Settings

We've purposefully kept editor configuration files out of this repo in order to keep it as simple and environment agnostic as possible. However, we recommend using the following settings:

- indent using spaces
- set tab width to 2 spaces (keeps the matrices in source code aligned)

Code Style

Please (do your best to) stick to Google's C++ style guide.

Project Instructions and Rubric

Note: regardless of the changes you make, your project must be buildable using cmake and make!

More information is only accessible by people who are already enrolled in Term 2 of CarND. If you are enrolled, see the project page for instructions and the project rubric.

Hints!

- You don't have to follow this directory structure, but if you do, your work will span all of the .cpp files here. Keep an eye out for TODOs.

Call for IDE Profiles Pull Requests

Help your fellow students!

We decided to create Makefiles with cmake to keep this project as platform agnostic as possible. Similarly, we omitted IDE profiles in order to we ensure that students don't feel pressured to use one IDE or another.

However! I'd love to help people get up and running with their IDEs of choice. If you've created a profile for an IDE that you think other students would appreciate, we'd love to have you add the requisite profile files and instructions to ide_profiles/. For example if you wanted to add a VS Code profile, you'd add:

- /ide_profiles/vscode/.vscode
- /ide_profiles/vscode/README.md

The README should explain what the profile does, how to take advantage of it, and how to install it.

Frankly, I've never been involved in a project with multiple IDE profiles before. I believe the best way to handle this would be to keep them out of the repo root to avoid clutter. My expectation is that most profiles will include instructions to copy files to a new location to get picked up by the IDE, but that's just a guess.

One last note here: regardless of the IDE used, every submitted project must still be compilable with cmake and make./

How to write a README

A well written README file can enhance your project and portfolio. Develop your abilities to create professional README files by completing this free course.