udacity / CarND-Kidnapped-Vehicle-Project

Watch 14 | Star 27 | Fork 360

Code | Issues 4 | Pull requests 8 | Projects 0 | Wiki | Insights

No description, website, or topics provided.

32 commits | 1 branch | 0 releases | 3 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

awbrown90 committed on GitHub Update particle_filter.cpp ... Latest commit 5f22ac5 on May 24

data	feat: updating visualizer to use c++ uwebsocketio	3 months ago
src	Update particle_filter.cpp	2 months ago
.gitignore	revise folder structure.	4 months ago
CMakeLists.txt	Merge branch 'master' into visualizer	2 months ago
README.md	Merge branch 'master' into visualizer	2 months ago
build.sh	add project files	4 months ago
clean.sh	add project files	4 months ago
cmakepatch.txt	Add files via upload	2 months ago
install-mac.sh	feat: adding the uwebsocketio installer scripts	3 months ago
install-ubuntu.sh	Add dependency installation: libssl-dev for Ubuntu	2 months ago
run.sh	Update run.sh	2 months ago

README.md

Overview

This repository contains all the code needed to complete the final project for the Localization course in Udacity's Self-Driving Car Nanodegree.

Submission

All you will submit is your completed version of `particle_filter.cpp`, which is located in the `src` directory. You should probably do a `git pull` before submitting to verify that your project passes the most up-to-date version of the grading code (there are some parameters in `src/main.cpp` which govern the requirements on accuracy and run time.)

Project Introduction

Your robot has been kidnapped and transported to a new location! Luckily it has a map of this location, a (noisy) GPS estimate of its initial location, and lots of (noisy) sensor and control data.

In this project you will implement a 2 dimensional particle filter in C++. Your particle filter will be given a map and some initial localization information (analogous to what a GPS would provide). At each time step your filter will also get observation and control data.

Running the Code

This project involves the Term 2 Simulator which can be downloaded here

This repository includes two files that can be used to set up and intall uWebSocketIO for either Linux or Mac systems. For windows you can use either Docker, VMware, or even Windows 10 Bash on Ubuntu to install uWebSocketIO.

Once the install for uWebSocketIO is complete, the main program can be built and ran by doing the following from the project top directory.

mkdir build cd build cmake .. make ./particle_filter

Note that the programs that need to be written to accomplish the project are src/particle_filter.cpp, and particle_filter.h

The program main.cpp has already been filled out, but feel free to modify it.

Here is the main protcol that main.cpp uses for uWebSocketIO in communicating with the simulator.

INPUT: values provided by the simulator to the c++ program

// sense noisy position data from the simulator

["sense_x"]

["sense_y"]

["sense_theta"]

// get the previous velocity and yaw rate to predict the particle's transitioned state

["previous_velocity"]

["previous_yawrate"]

// receive noisy observation data from the simulator, in a respective list of x/y values

["sense_observations_x"]

["sense_observations_y"]

OUTPUT: values provided by the c++ program to the simulator

// best particle values used for calculating the error evaluation

["best_particle_x"]

["best_particle_y"]

["best_particle_theta"]

//Optional message data used for debugging particle's sensing and associations

// for respective (x,y) sensed positions ID label

["best_particle_associations"]

// for respective (x,y) sensed positions

["best_particle_sense_x"] <= list of sensed x positions

["best_particle_sense_y"] <= list of sensed y positions

Your job is to build out the methods in `particle_filter.cpp` until the simulator output says:

```
Success! Your particle filter passed!
```

Implementing the Particle Filter

The directory structure of this repository is as follows:

```
root
|   build.sh
|   clean.sh
|   CMakeLists.txt
|   README.md
|   run.sh
|
|___data
|   |
|   |   map_data.txt
|
|
|___src
    |   helper_functions.h
    |   main.cpp
    |   map.h
    |   particle_filter.cpp
    |   particle_filter.h
```

The only file you should modify is `particle_filter.cpp` in the `src` directory. The file contains the scaffolding of a `ParticleFilter` class and some associated methods. Read through the code, the comments, and the header file `particle_filter.h` to get a sense for what this code is expected to do.

If you are interested, take a look at `src/main.cpp` as well. This file contains the code that will actually be running your particle filter and calling the associated methods.

Inputs to the Particle Filter

You can find the inputs to the particle filter in the `data` directory.

The Map*

`map_data.txt` includes the position of landmarks (in meters) on an arbitrary Cartesian coordinate system. Each row has three columns

1. x position
2. y position
3. landmark id

All other data the simulator provides, such as observations and controls.

> - Map data provided by 3D Mapping Solutions GmbH.

Success Criteria

If your particle filter passes the current grading code in the simulator (you can make sure you have the current version at any time by doing a `git pull`), then you should pass!

The things the grading code is looking for are:

1. **Accuracy**: your particle filter should localize vehicle position and yaw to within the values specified in the parameters `max_translation_error` and `max_yaw_error` in `src/main.cpp` .
2. **Performance**: your particle filter should complete execution within the time of 100 seconds.