UDACITY

‹ Return to "Self-Driving Car Engineer" in the classroom

DISCUSS ON STUDENT HUB

Model Predictive Control (MPC)

REVIEW

CODE REVIEW 3

HISTORY

▼ src/main.cpp 2

```
#include <math.h>
#include <uWS/uWS.h>
#include <chrono>
#include <iostream>
#include <string>
#include <thread>
#include <vector>
#include "Eigen-3.3/Eigen/Core"
#include "Eigen-3.3/Eigen/QR"
#include "helpers.h"
#include "json.hpp"
#include "MPC.h"

// for convenience
using nlohmann::json;
using std::string;
using std::vector;
using Eigen::VectorXd;

// For converting back and forth between radians and degrees.
constexpr double pi() { return M_PI; }
double deg2rad(double x) { return x * pi() / 180; }
double rad2deg(double x) { return x * 180 / pi(); }

const double Lf = 2.67;

int main() {
  uWS::Hub h;

  // MPC is initialized here!
```

```
31  MPC mpc;
32
33  h.onMessage([&mpc](uWS::WebSocket<uWS::SERVER> ws, char *data, size_t length,
34                     uWS::OpCode opCode) {
35    // "42" at the start of the message means there's a websocket message event.
36    // The 4 signifies a websocket message
37    // The 2 signifies a websocket event
38    string sdata = string(data).substr(0, length);
39    std::cout << sdata << std::endl;
40    if (sdata.size() > 2 && sdata[0] == '4' && sdata[1] == '2') {
41      string s = hasData(sdata);
42      if (s != "") {
43        auto j = json::parse(s);
44        string event = j[0].get<string>();
45        if (event == "telemetry") {
46          // j[1] is the data JSON object
47          vector<double> ptsx = j[1]["ptsx"];
48          vector<double> ptsy = j[1]["ptsy"];
49          double px = j[1]["x"];
50          double py = j[1]["y"];
51          double psi = j[1]["psi"];
52          double v = j[1]["speed"];
53          double delta= j[1]["steering_angle"];
54
55          /**
56           * TODO: Calculate steering angle and throttle using MPC.
57           * Both are in between [-1, 1].
58           */
59          // Transform the waypoint from global coordinates to car coordinate syst
60
61          VectorXd ptsx_trans(ptsx.size());
62          VectorXd ptsy_trans(ptsy.size());
63          for(unsigned int i = 0; i<ptsx.size(); i++){
64            // Move the coordinate system
65            double dx = ptsx[i] - px;
66            double dy = ptsy[i] - py;
67            // Rotate new coordinate system
68            ptsx_trans[i] = ( dx * cos(psi) + dy * sin(psi) );
69            ptsy_trans[i] = ( -dx * sin(psi) + dy * cos(psi) );
```

AWESOME

Nice work in transforming the waypoint to vehicle coordinates!

```
70          }
71
72           //fit a 3rd orden polynomial to the x and y coordinates
73          auto coeffs = polyfit(ptsx_trans, ptsy_trans, 3);
74
75          /**
76           * Try to compensate the effect of the latency. We are going to calculat
77           * the status of the car in the future
78           */
79          // Actuator latency in seconds
80          double latency = 0.1;
81
82          // Initial state
83          double x0 = 0;
84          double y0 = 0;
85          double psi0 = 0;
86          double v0 = v;
87          double cte0 = polyeval(coeffs, 0); // cte[t] = f(x[t-1]) - 0 + 0
            double epsi0 = -atan(coeffs[1]);  // epsi[t] = 0 - psides[t-1] + 0
```

```
88
90
91          // Initial state modified due latency
92          double x_d = x0 + ( v * cos(psi0) * latency );  // x_[t] = x[t-1] + v[t-
93          double y_d = y0 + ( v * sin(psi0) * latency ); // y_[t] = y[t-1] + v[t-1
94          double psi_d = psi0 - ( v * delta * latency / Lf ); // psi_[t] = psi[t-1
95          double v_d = v0;  // same speed because i can't  tranform acceleration f
96          double cte_d = cte0 + ( v * sin(epsi0) * latency );
97          double epsi_d = epsi0 - ( v * atan(coeffs[1]) * latency / Lf ); // epsi[
```

REQUIRED

atan(coeffs[1]) is angle derived froom the given trajectory, steering angle should be used instead to re

```
98
99
100         // Run MPC
101         Eigen::VectorXd state(6);
102         state << x_d, y_d, psi_d, v_d, cte_d, epsi_d;
103         auto vars = mpc.Solve(state, coeffs);
104
105         double steer_value = vars[0];
106         double throttle_value = vars[1];
107
108         json msgJson;
109         // NOTE: Remember to divide by deg2rad(25) before you send the
110         //   steering value back. Otherwise the values will be in between
111         //   [-deg2rad(25), deg2rad(25] instead of [-1, 1].
112         msgJson["steering_angle"] = steer_value/deg2rad(25);
113         msgJson["throttle"] = throttle_value;
114
115         // Display the MPC predicted trajectory  in green
116         vector<double> mpc_x_vals;
117         vector<double> mpc_y_vals;
118
119         /**
120          * TODO: add (x,y) points to list here, points are in reference to
121          *   the vehicle's coordinate system the points in the simulator are
122          *   connected by a Green line
123          */
124         for (unsigned int i = 2 ; i < vars.size(); i += 2){
125           mpc_x_vals.push_back(vars[i]);
126           mpc_y_vals.push_back(vars[i + 1]);
127         }
128
129         msgJson["mpc_x"] = mpc_x_vals;
130         msgJson["mpc_y"] = mpc_y_vals;
131
132         // Display the waypoints/reference line in yellow
133         // I use the same ptsx values but fitted the third order polynomial
134         vector<double> next_x_vals;
135         vector<double> next_y_vals;
136         for (unsigned int i = 0 ; i < ptsx_trans.size(); i++){
137           next_x_vals.push_back(ptsx_trans[i]);
138           next_y_vals.push_back(polyeval(coeffs, ptsx_trans[i]));
139         }
140
141         /**
142          * TODO: add (x,y) points to list here, points are in reference to
143          *   the vehicle's coordinate system the points in the simulator are
144          *   connected by a Yellow line
145          */
```

```
          msgJson["next_x"] = next_x_vals;
          msgJson["next_y"] = next_y_vals;


          auto msg = "42[\"steer\"," + msgJson.dump() + "]";
          std::cout << msg << std::endl;
          // Latency
          // The purpose is to mimic real driving conditions where
          //   the car does actuate the commands instantly.
          //
          // Feel free to play around with this value but should be to drive
          //   around the track with 100ms latency.
          //
          // NOTE: REMEMBER TO SET THIS TO 100 MILLISECONDS BEFORE SUBMITTING.
          std::this_thread::sleep_for(std::chrono::milliseconds(100));
          ws.send(msg.data(), msg.length(), uWS::OpCode::TEXT);
        }  // end "telemetry" if
      } else {
        // Manual driving
        std::string msg = "42[\"manual\",{}]";
        ws.send(msg.data(), msg.length(), uWS::OpCode::TEXT);
      }
    }  // end websocket if
  }); // end h.onMessage

  h.onConnection([&h](uWS::WebSocket<uWS::SERVER> ws, uWS::HttpRequest req) {
    std::cout << "Connected!!!" << std::endl;
  });

  h.onDisconnection([&h](uWS::WebSocket<uWS::SERVER> ws, int code,
                         char *message, size_t length) {
    ws.close();
    std::cout << "Disconnected" << std::endl;
  });

  int port = 4567;
  if (h.listen(port)) {
    std::cout << "Listening to port " << port << std::endl;
  } else {
    std::cout << "Failed to listen to port" << std::endl;
    return -1;
  }

  h.run();
}
```

▸ src/MPC.cpp 1

▸ writeup.md

▸ src/helpers.h

▸ src/MPC.h

▸ set_git.sh

▸ scotch5.rb

- scotch.rb
- output/cmake_install.cmake
- output/Makefile
- output/CMakeFiles/progress.marks
- output/CMakeFiles/mpc.dir/progress.make
- output/CMakeFiles/mpc.dir/link.txt
- output/CMakeFiles/mpc.dir/flags.make
- output/CMakeFiles/mpc.dir/depend.make
- output/CMakeFiles/mpc.dir/depend.internal
- output/CMakeFiles/mpc.dir/cmake_clean.cmake
- output/CMakeFiles/mpc.dir/build.make
- output/CMakeFiles/mpc.dir/DependInfo.cmake
- output/CMakeFiles/mpc.dir/CXX.includecache
- output/CMakeFiles/feature_tests.cxx
- output/CMakeFiles/feature_tests.c
- output/CMakeFiles/feature_tests.bin
- output/CMakeFiles/cmake.check_cache
- output/CMakeFiles/TargetDirectories.txt
- output/CMakeFiles/Makefile2
- output/CMakeFiles/Makefile.cmake
- output/CMakeFiles/CMakeOutput.log
- output/CMakeFiles/CMakeDirectoryInformation.cmake

- output/CMakeFiles/3.5.1/CompilerIdCXX/a.out
- output/CMakeFiles/3.5.1/CompilerIdCXX/CMakeCXXCompilerId.cpp
- output/CMakeFiles/3.5.1/CompilerIdC/a.out
- output/CMakeFiles/3.5.1/CompilerIdC/CMakeCCompilerId.c
- output/CMakeFiles/3.5.1/CMakeSystem.cmake
- output/CMakeFiles/3.5.1/CMakeDetermineCompilerABI_CXX.bin
- output/CMakeFiles/3.5.1/CMakeDetermineCompilerABI_C.bin
- output/CMakeFiles/3.5.1/CMakeCXXCompiler.cmake
- output/CMakeFiles/3.5.1/CMakeCCompiler.cmake
- output/CMakeCache.txt
- mumps.rb
- lake_track_waypoints.csv
- ipopt.rb
- install_ipopt.sh
- install_Ipopt_CppAD.md
- install-ubuntu.sh
- install-mac.sh
- docker-compose.yml
- cmakepatch.txt
- README.md
- LICENSE

- Dockerfile
- DATA.md
- CODEOWNERS
- CMakeLists.txt
- .gitignore

RETURN TO PATH

Rate this review