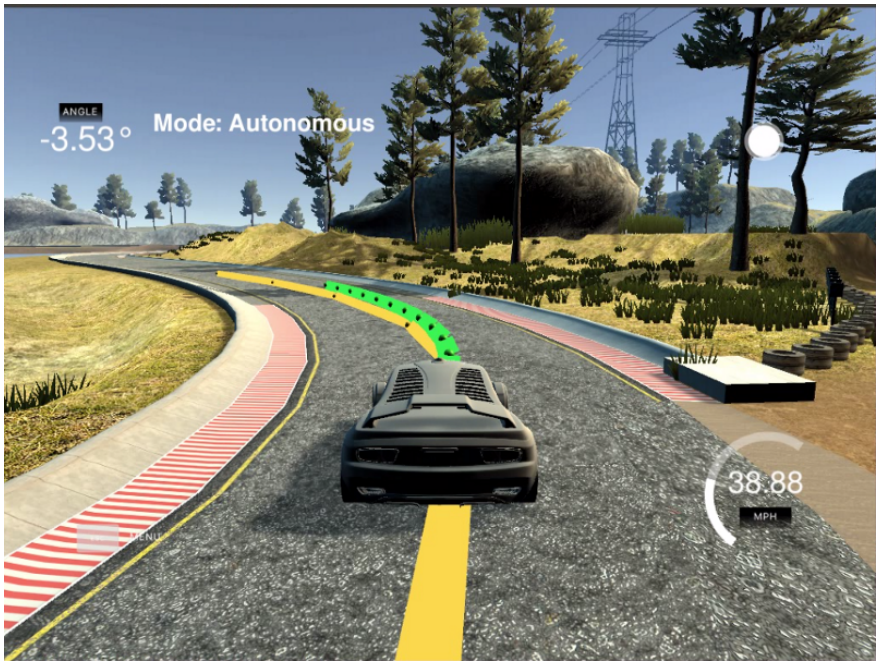
デビッドシルバー フォローする
私は自家用車が大好きで、私は@Udacityで仕事をしています！私たちと一緒に自走車を構築する方法を学ぶ：https://udacity.com/drive
2017年7月10日 · 4分読み取り

5人の異なるUDACITY学生用コントローラ



Udacity自己運転車技術者Nanodegreeプログラムの第六月は、制御について学生に教える。コントロールとは、ステアリングを実際に回すか、ペダルを踏んで軌道をたどる方法です。この作業を実行するアルゴリズムは、「コントローラ」と呼ばれます。

車載アプリケーション用の最も一般的なコントローラーの2つは、比例積分微分（PID）コントローラーとモデル予測コントローラー（MPC）です。これらはUdacityプログラムで教える2つのコントローラです。

ここでは、Udacityの学生がシミュレータの周りにUdacityの自転車を運転するコントローラを作るために取った5つの異なるアプローチがあります！

PIDコントローラ、自家用車

アンドレイ・グリュシコ

AndreyのYouTubeのビデオでは、PIDコントローラーが自動的にハイパーパラメーターを学習すると述べています。Andreyはトラック周りに車を何度も走らせ、自動的にパラメータを調整するために座標降下（セバスチャンのTwiddleアルゴリズムの正式名称）を使用したようです。

> *「C ++で自動的に学習されたハイパーパラメータを備えたPIDコントローラを実装することで、シミュレータのスクラッチから自動車を学ぶことができます。*

予測制御モデルを用いた自律運転

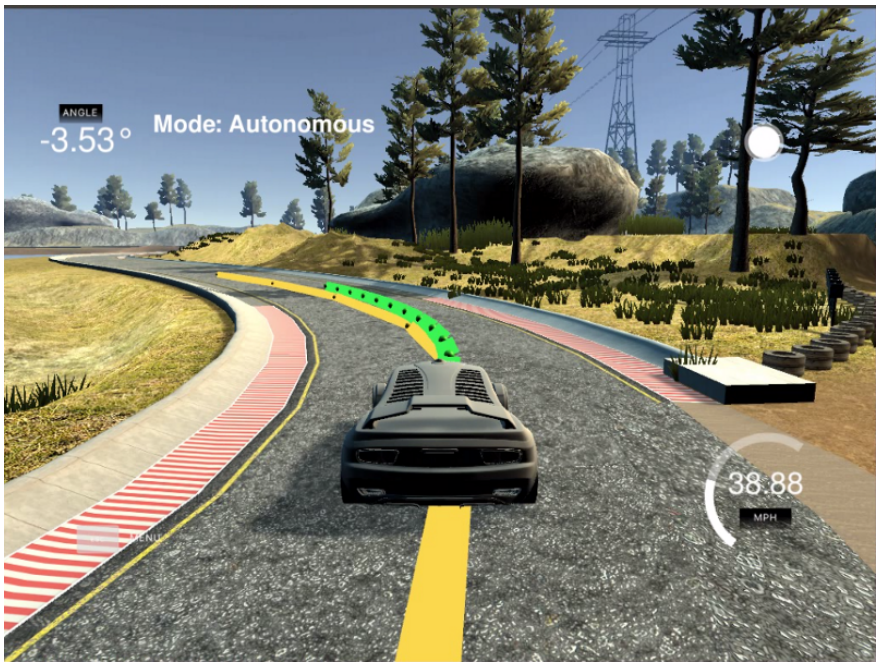
Anupriya Chhabra

Anupriyaは、彼女がモデル予測コントローラを開発する際に遭遇したいくつかの複雑さと、彼女がそれらをどのように克服したかについて語っています。これらの多くは、実際の車両にコントローラを配備する際に自律的な車両エンジニアが見つけるものと似ています。

> *"このプロジェクトはまた、アクチュエーター入力の適用中に発生する可能性のある実際のレイテンシを考慮しています。これをシミュレートするために、プロジェクトのメインスレッドは100msの間スリープしてからシミュレータにアクチュエーションを送信します。これを考慮に入れて、シミュレータに操作を戻すには、次のステップの実際の操作と、私の場合は0.1秒（100ms）のdt後の次の予測される起動を使用します。これらの2つのアクチュエーションの合計を送信すると、モデルは次のアクチュエーションを積極的に適用し、100msのレイテンシを処理します。*

自走車のステアリング制御

Priya Dwivedi

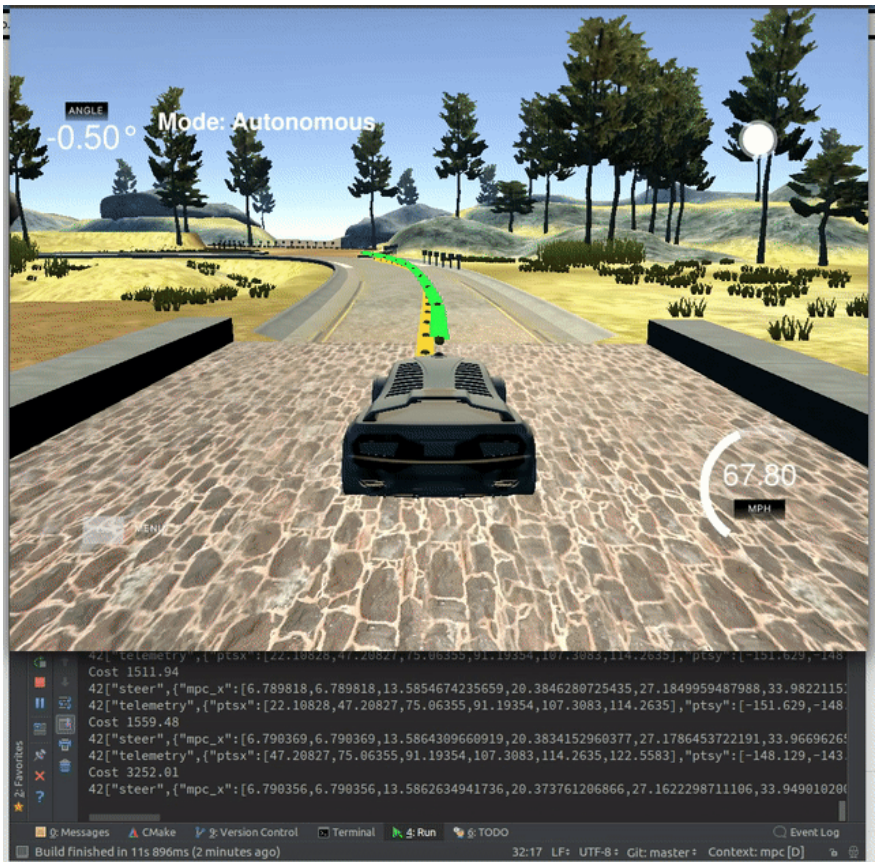


Priyaは、彼女が彼女のモデル予測コントローラーをどのように構築したかを徹底的に見てきました。MPCがどのように動作しているかについて、行ごとに細分化することに興味があるなら、これは素晴らしい読書です。

> *理想的なステアリング角度とスロットルを推定するために、理想状態からの新しい状態の誤差、つまり実際の軌道と維持したい速度と方向を推定し、この誤差を最小限に抑えるためにIpoptソルバーを使用します。これは、所望の軌道の誤差を最小にするステアリングとスロットルを選択するのに役立ちます。*

自己運転自動車エンジニアダイアリー - 10

アンドリュー・ウィルキー



Andrewは、PIDとMPCを比較し、Nanodegreeプログラムの第2項を簡単に見直します。彼は、あなたがコントローラを構築することができる忠実度の様々なレベルを要約する素晴らしい仕事をしています。

> *「PIDコントローラは、比例的に（ステアリングを補正するのが難しい）、差異的に（基準に戻るまでに徐々に戻っていく）、一体的に（ホイールの位置ずれを許容しながら）カーブ（ロボット）クロストラックエラー（CTE）。これはコード化が簡単で、実行するのに安価であり（計算的に）、何か動作するためのチューニング作業をほとんど必要としません。欠点は、車が不規則に動いて、作動待ち時間（コマンド送信と物理的な起動の間の遅延）を正確に処理できないことです。*

UDACITY自己駆動車Nanodegreeプロジェクト10 - モデル予測制御

ジェレミーシャノン

I enjoy the musical selections Jeremy picks to underscore his project submissions, and this one is a lot of fun. The blog post also provides a nice perspective of what it's like to complete the Udacity MPC project as a student.

> *"After some debugging and tuning the cost function, my car was making its way around the track. It was time to tear it all down by adding the latency —and that's just what happened. My approach to dealing with it was twofold (not counting simply limiting the speed): the original kinematic equations depend upon the actuations from the previous time step, but with a delay of 100ms (which happened to be my time step interval) the actuations are applied another time step later, so I altered the equations to account for this."*