John Alsterda
CS 221: AI
25 Oct 2018

Online Wheel Alignment via Machine Learning - Project Proposal

Introduction

Vehicles such as cars and trucks undergo periodic maintenance to calibrate the alignment of their wheels. The manual procedure typically consists of adjusting the angles of each wheel, so they point forward to 0° when the steering wheel is at 0°. As illustrated to the right, a fixture mounted to the wheel helps a technician correct the alignment. Wheels typically become misaligned after striking obstacles such as curbs or pot-holes. A vehicle with misaligned wheels will 'pull' left or right and wear down their tires more quickly and unevenly [1].

On most vehicles, alignment is adjusted mechanically; a fine-adjustment screw is turned to set its default angle. On the contrary, for X1 -- the Dynamic Design Lab's student-built 4-wheel-steering automated research platform, alignment is set in software. We find the zero angles manually, and record the observed motor encoder ticks to X1's parameter script. For my CS 221 project, I plan to perform this tedious job automatically online using AI / ML.

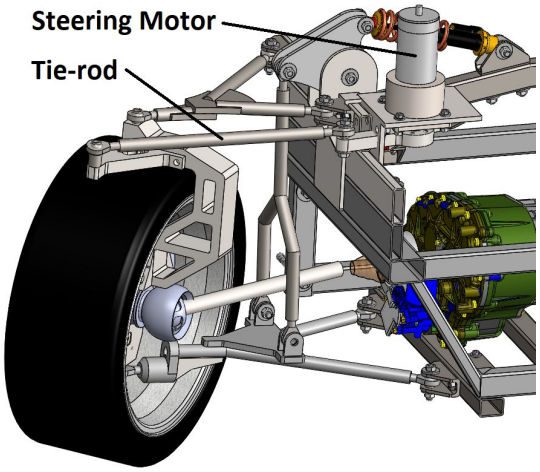
Model

At a high level, as X1 drives, I plan to compare its change in state to the change predicted by a model, calculating an error Δx:

$$\Delta x = x_{+}^{model} - x_{+}^{measured}$$

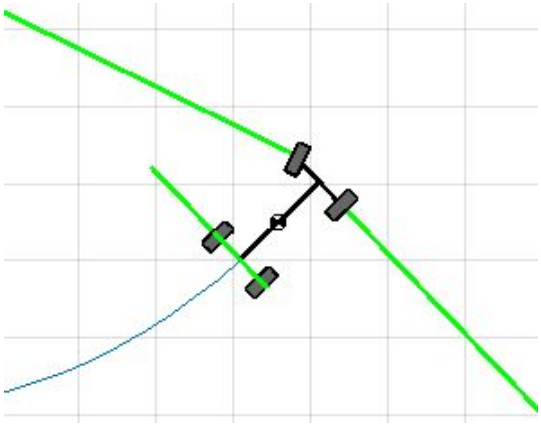
where $x_{+}^{measured}$ is the vehicle state measured by X1's onboard GPS/INS suite, including positions, velocities, and accelerations. x_{+}^{model} is the vehicle state predicted by a model:

$$x_{+}^{model} = f\left(x_{0}^{measured}, u, \Delta t \right).$$



$f()$ is a 4-wheel vehicle model which calculates the anticipated evolution of x_0 when command u is applied for Δt seconds. The command includes individual steering angles for each of X1's 4 independently steered wheels, illustrated at right. These state transitions will be treated as a Markov Chain, motivating the following ML strategies to search for true wheel angles and optimal alignment corrections.

At right, a snapshot is shown of the 4-wheel vehicle model with a significant 20° left misalignment in the front left wheel. The green lines show the lateral tire forces being generated at each wheel, causing the vehicle to pull left.



Algorithm

The first algorithm I plan to implement is Q-Learning. In Q-Learning, an optimal policy is generated by estimating the value of each state-action pair [2]. My states, actions, and rewards are:

- States: a correction angle $\boldsymbol{\Delta\delta}$ for each wheel. $S \in R^4$, possibly discretized as shown at right.
- Actions: choice of new correction angles, a $\boldsymbol{\Delta}$-$\boldsymbol{\Delta\delta}$ if you will. Possibly discretized, possibly finite choices of +¼, +0, -¼ degree for example.
- Reward: Penalty is assigned based on $|\boldsymbol{\Delta}x|$, motivating the algorithm to find the state which allows X1 to most closely follow the vehicle model f().

$\boldsymbol{\Delta\delta}_{FL}$	…	-¼	0	+¼	-½	…
$\boldsymbol{\Delta\delta}_{FR}$	…	-¼	0	+¼	-½	…
$\boldsymbol{\Delta\delta}_{RL}$	…	-¼	0	+¼	-½	…
$\boldsymbol{\Delta\delta}_{RR}$	…	-¼	0	+¼	-½	…

The algorithm will run until convergence, and each $\boldsymbol{\Delta\delta}$ will be regarded as the current misalignment angle for the corresponding wheel. The encoder tick for this angle can then to saved to X1's parameters for future experiments.

Interestingly, although Q-Learning is a model-free method, I'm still incorporating a model to calculate the reward. I'm interested to leverage the model to help my algorithm decide which sign the action should take. For example, if X1 should go straight but pulls left, then actions should favor right-trending corrections.

One challenging aspect of this task lies in the fact that multiple steering correction choices could yield similar model-following performance. For example, if the front wheels are both toe-in by the same amount, X1 will still move forward (but scrub at each wheel). This condition could be relieved by penalizing the motor current required to hold these competing steering angles.

Another potential algorithm is simple gradient descent on the cost function Δx^2 [3]. Substituting a linearized version of my model into this function could yield a convex function amenable to minimization.

Experiment

I plan to validate my algorithms in experiment on X1 before end-of-quarter. This project could lead to a significant improvement in X1 alignment and relieve us of the headache of manual alignment in the future.

References

1. "Wheel alignment." Wikipedia. https://en.wikipedia.org/wiki/Wheel_alignment
2. "Q-Learning." Wikipedia. https://en.wikipedia.org/wiki/Q-learning
3. "Gradient descent." Wikipedia. https://en.wikipedia.org/wiki/Gradient_descent