Checkpoint 1.1 Progress Report

Logan Zartman

Austin Atchley

1. Mathematical computation of our 1-D time optimal controller:

Let `Vo` be the velocity as reported by odometry. Let `Xo` be the distance travelled as computed by the distance between the initial odometry-reported location and the current odometry-reported location. Let `Dmax` be the maximum rate of deceleration. Let `Amax` be the maximum rate of acceleration. Let `Smax` be the maximum speed.

```
For each timestep:
    Let speed = norm(Vo)
    Let next_position = Xo + speed * timestep_duration
    Let time_to_stop = speed / Dmax
    Let stop_position = next_position + (speed * time_to_stop) + (1/2 * Dmax * time_to_stop^2)

    If stop_position > target_position:
        Let remaining_distance = max(0, target_position - position)
        Let deceleration = -speed^2 / (2 * remaining_distance)
        output_speed = max(0, speed + deceleration * timestep_duration)
    Else:
        output_speed = min(Smax, output_speed + Amax * timestep_duration)
```

2. Code organization:

Most of the code we wrote for this project resides in `navigation/navigation.h` and `navigation/navigation.cc`. We began by sending a basic `AckermannCurvatureDriveMsg` to the car. We proceeded by storing state every time the car's callbacks were called. We used the data to calculate the distance we had traveled, and implemented the dead reckoning approach. This was all done in the `Navigation::Run` method. Next, we began to use the odometry data to get a better idea of where the car is. This made use of the `Navigation::UpdateOdometry` callback. We used this data to implement the approach outlined on the slides in class.

We predict where the car will be on the next timestep. If the car is not at full velocity and far enough away from the target destination, we accelerate at maximum acceleration. If the car would be past the deceleration target, we decelerate at maximum deceleration. Thus, the car stays under its limits and satisfies the 1D time-optimal control problem.

We use basic kinematic equations to make these calculations. In particular, we solve `x=vt+(1/2)at^2` to compute distances and `v^2=v0^2+2aΔx` to compute deceleration. We take the velocity we've calculated with these equations and send it to the car with `ros::Publisher::publish`.

Parameter tuning:

Adjusted steering offset by doing a rough binary search and visually confirming driving straightness.

3. Challenges faced:

The main challenge we faced was reconciling the speed we calculated with the speed we received from the odometer readings. We weren't exactly sure how to combine the two pieces of data at first, but we came up with a solution that seems to work well enough so that our car lands within +-0.05m over 2m in the simulation. We will continue to improve the calculations in the coming checkpoints.

4. Contributions of each team member:

Austin - Initial structure, dead reckoning, and external tooling setup for the rest of the semester

Logan - Kinematic calculations, using odometry data to track position and velocity

5. GitHub Repo

https://github.com/austinatchley/F1-10-Autonomous-Driving

6. Video Demo

https://drive.google.com/file/d/1jy1_ZjBqPbRSO6X4pS_B2Ihz2VXAvf-o/view?usp=sharing

Demo shows car moving 2m (one grid tile.)

Simulated odometry indicated distance error consistently less than 1cm.

7. Video of code on real car

Demos show car moving 2m.

0.5m/s max speed

https://drive.google.com/open?id=1eaX8Jz-jSqix1y5RraKjuk0GFxDMnGIp

Run	Encoder distance
1	2.0269m
2	2.0159m
3	2.0143m

Average error: 0.019m

1.0m/s max speed

https://drive.google.com/open?id=1edVffU4A8XDwHRRnZ1pdLsrAussCRl6K

8. Future improvements:

We would like to have `_toc_speed` and `speed` readings converge. That is to say, we want to take our two separate speeds and get more accurate results as a result of having more data. Right now, we aren't sure if our method is the most correct way of combining the two readings. Additionally, we considered adjusting the velocity output using something like to a PID controller.

Obviously, we would like the car to be able to go in other directions. This is for the next checkpoint.