Motion Script

// This is a preliminary script. It does not have any activities or knowledge checks.

Intro

Course Description

A look at how the AV turns perception inputs into motion.

Learning Objectives

By the end of this course, you should be able to:

- Understand how we go from sensor data to sending commands to the car
- Connect the dots on the overall stack and where your work fits in
- Identify the people, teams and overall structure of AV Engineering
- Identify the responsibilities of the major AV Eng teams

Course Duration: 30 min

To begin, click the first lesson below, or the "Start Course" button above.

Happy Learning!

Introduction

This course is about taking the AV from standing still to navigating obstacles. It will be followed by a live review, so please note any questions you have for the live session.

Before the AV even gets in motion we have to start with localization.

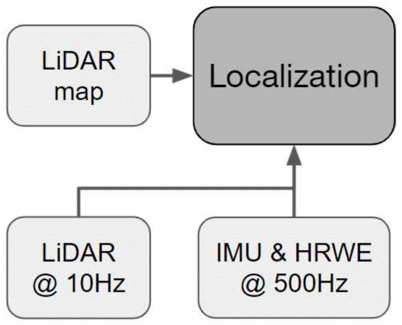
Localization

When we place a vehicle on the road, it needs to know where it is in relation to the world. For this we use localization which allows the AV to know where it is.

[GRAPHIC: Localization - The process of orienting the vehicle on a map]

We use localization to see where the AV is on the map, ground sensor readings, and to decide on what routes to take.

This is an iterative process that uses a static map, lidar returns, accelerations provided by the inertial measurement unit (IMU), and displacements provided by the high resolution wheel encoders (HRWE) to localize the vehicle in the world.

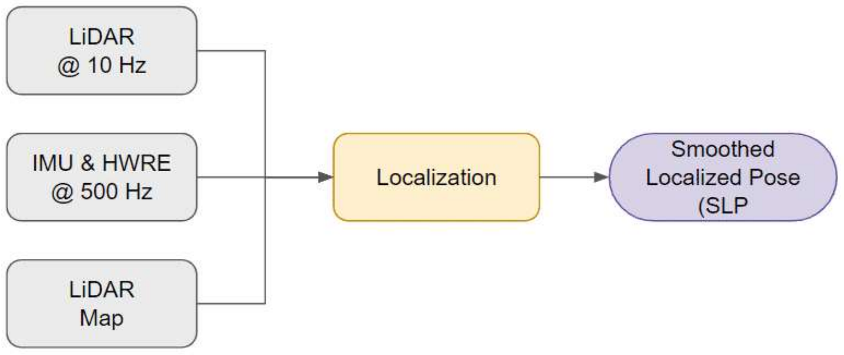


[GRAPHIC: High Resolution Wheel Encoders]

How does localization occur?

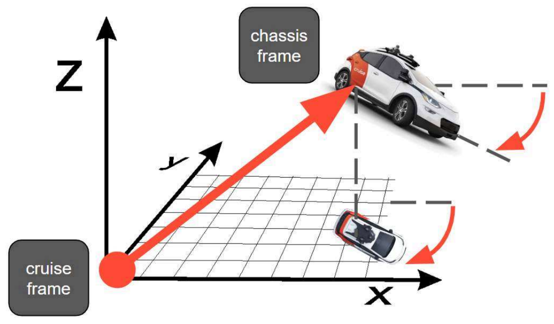
The AV starts in a Cruise garage,t from a known localization; as it starts to move, the AV updates its localization based on acceleration and heading estimates provided by the Inertial Measurement Unit (IMU), and based on displacement estimates provided by the High Resolution Wheel Encoders (HWRE).

As the AV drives out to the street, it finds objects with high Lidar returns such as tall walls and road markings. We combine real-time observations with the existing map to get an estimate of the position of the AV in the world. This is what we call the Smoothed Localized Pose (SLP); and it is the closest thing that we have to a ground truth.



Localization is expressed as a 3D position and orientation. It uses our classic X,Y, Z positions, and it uses the three angles of pitch, role, and yaw. All of this information is required in order to maintain the fine control that we need.

The rotations of the vehicle do not matter for many of the functions in the stack (such as routing), but for controls these are really important.



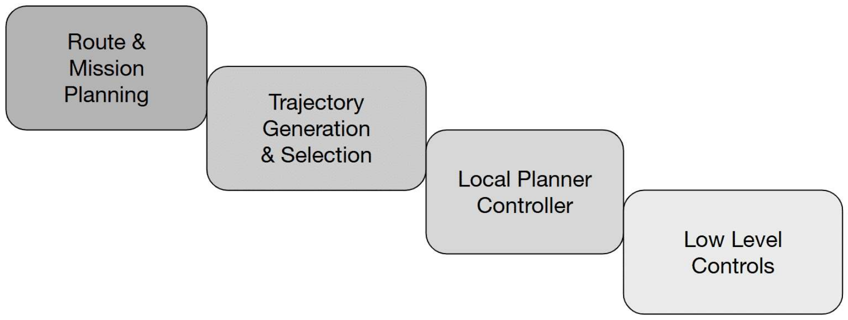
For example, we want to be able to compensate for road bank and road pitch (or grade), in order to provide the smoothest ride to the passenger.

We use a local reference frame for every market that we are in. Basically, this vector reports the position of the middle of the AV's rear axle of the AV with respect to the origin of the coordinate frame. Each market has its own reference frame. To localize points within the AV itself, we use the chassis frame reference. This starts in the middle of the rear axle. This allows us to localize the cameras and the sensors.

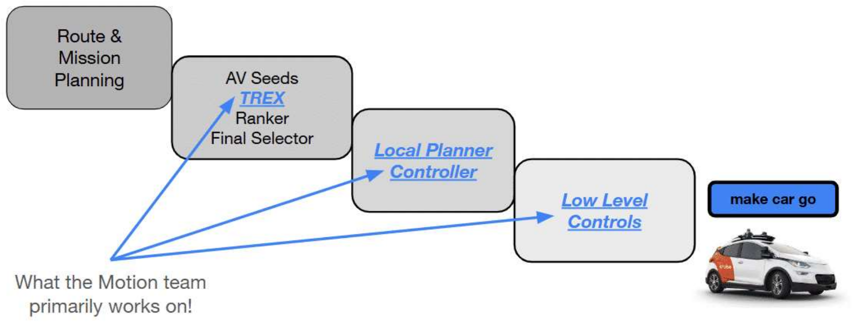
The 3D orientation is technically expressed as a quaternion (4 numbers), not as 3 Euler angles.

Overview of the Planning & Controls Stack

Here is a look at our stack. As you move down and to the right, the corresponding spatial and temporal domains increase in resolution.



The Motion Team deals with the last three areas.



Route Planning is similar in concept to Google maps. It provides the shortest path to the current destination.

Next, Maneuver Planning; given multiple trajectory options, map features, tracked cars, and peds, etc., maneuver planning chooses an instantaneous trajectory for execution.

Motion Planning and Control; here we process the trajectory from maneuver planning with vehicle dynamics considerations and ultimately actuate the AV. The terms L1 and L4 aren't used that often, but L2 and L3 are common, e.g. "level 3 planning meganode." Myriads of feedback loops from lower levels up to higher levels.

TREX

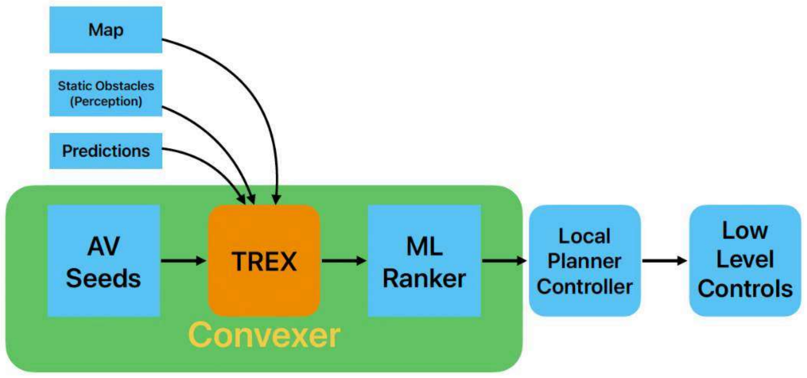
TREX stands for trajectory refinement and expansion. The goal of this is to refine and expand AV Seeds trajectories.

AV Seeds are trajectories generated by machine learning models, based on human driving trajectories. Trajectories are based on the current context (NPC positions and displacements; obstacles in the environment), and the trajectories generated offer no guarantees in terms of kinematic consistency, or even that every obstacle on the road is avoided.

TREX creates a guardrail, generating trajectories avoiding collisions and other hazardous events, based on the available information. Solutions guarantee kinematic consistency and obstacle avoidance explicitly, ensuring the safety of the AV on the road.

As shown here (below), TREX consumes information like maps, static obstacles, and predictions.

We also implemented a machine learning model, called L-TPG, which provides better initial guesses to the TREX module. It uses all of the features like initial curvature, actual speed and velocity, x- y- locations, in order to determine the best starting point for the optimization problem.



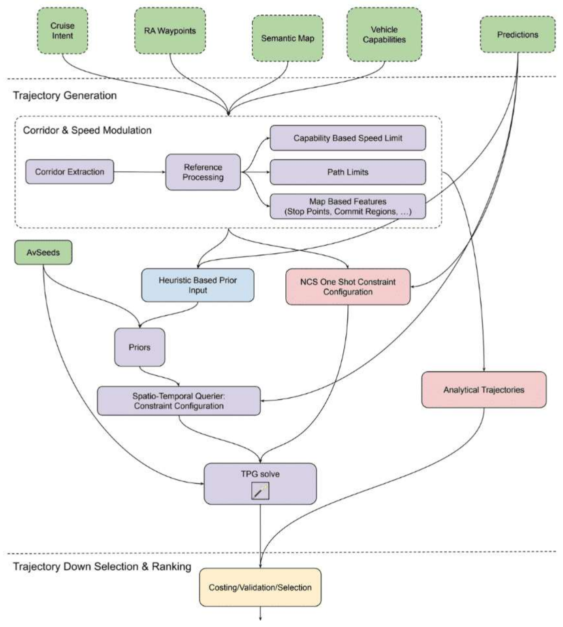
Finding a good starting point is important, for computational efficiency reasons; good starting points also can place the underlying optimizer upstream of higher quality solutions by avoiding local minima.

Once we have the trajectory from TREX, we send it to the ML ranker which helps us with the trajectory selection. It uses the same encodings as our learning models to pick the best trajectories out of 60 candidate trajectories to execute on.

This diagram (below) gives a look at how things work within TREX.

The green boxes are inputs. The Cruise intent is where the AV wants to go.

RA waypoints - If there is something that the AV cannot handle on the road, then our Remote Assistance team will draw some waypoints for it to go around.

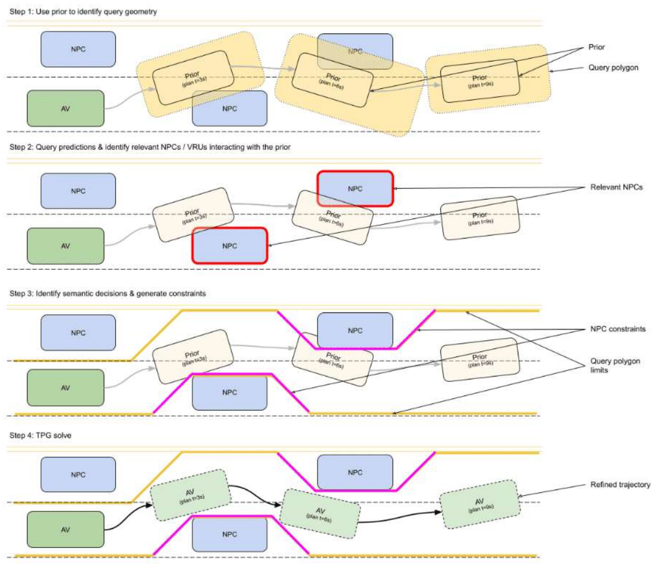

Cruise Intent
RA Waypoints
Semantic Map
Vehicle Capabilities
Predictions
Trajectory Generation
Corridor & Speed Modulation
Corridor Extraction
Reference Processing
Capability Based Speed Limit
Path Limits
Map Based Features (Stop Points, Commit Regions, ...)
AvSeeds
Heuristic Based Prior Input
NCS One Shot Constraint Configuration
Priors
Spatio-Temporal Querier: Constraint Configuration
Analytical Trajectories
TPG solve
Trajectory Down Selection & Ranking
Costing/Validation/Selection

Spatial Temporal Querier

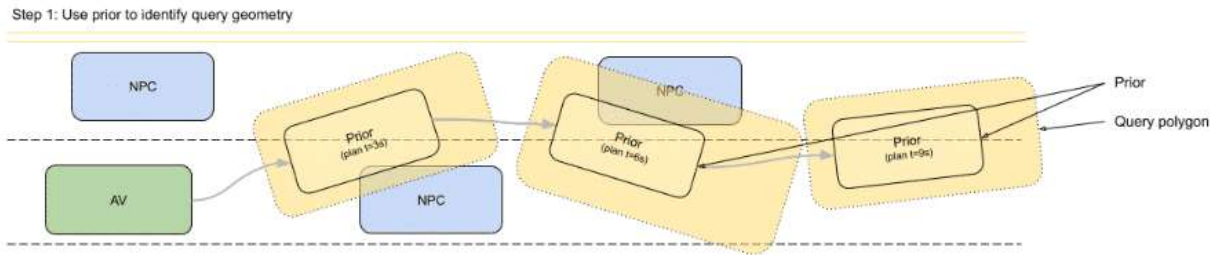
TREX relies on priors. Priors can be from the map or AV seeds. All of this goes through the corridor to determine the most relevant obstacles that our optimizer should avoid.

Once that is figured out we generate mathematical constraints using a non-linear software called TPG that provides an optimized directory regarding cost functions and constraints that are applied to the problem instance.

We have a way to detect other relevant NPCs. As shown here, it is the two highlighted in red. We determine that we have a right constraint so we have to bias to the right. And then we see that there is a left constraint.

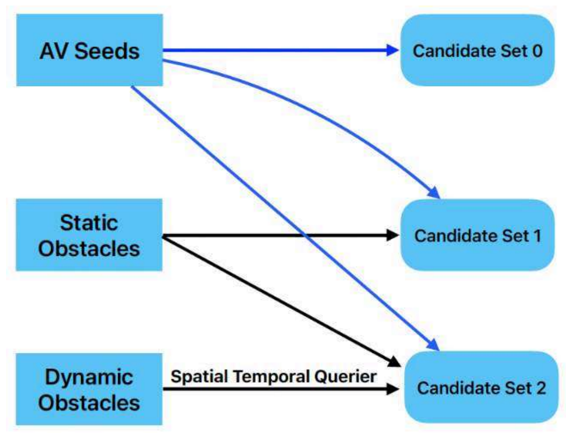


Based on this, we generate the lateral obstacle avoidance constraint. Then we figure out the best trajectory that is comfortable and safe. This is called the refined trajectory.



Some trajectories only consume AV seeds, some consider static obstacles, and others consider the dynamic obstacles.

We have three sets of trajectories to cover the diversity of the trajectories.



Candidate Sets

This diagram shows the three different sets of trajectories, referred to as Set 0, Set 1, and Set 2:

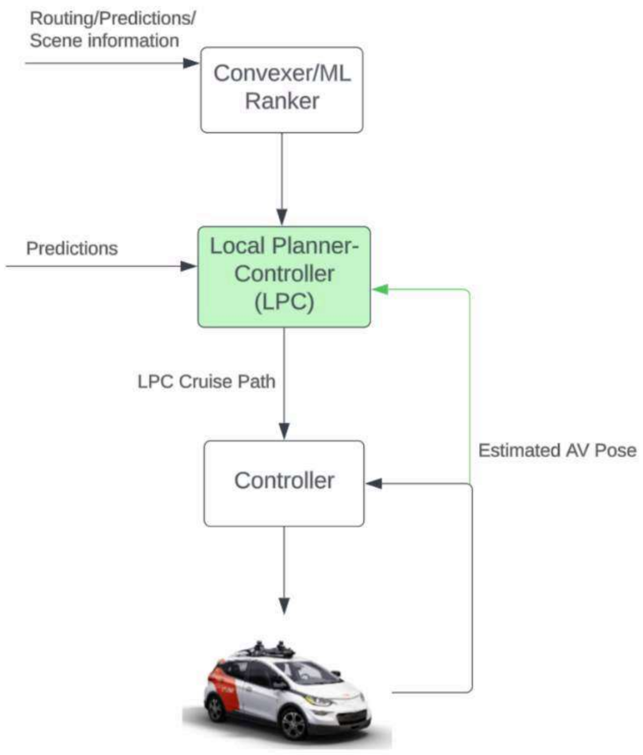
- Set 0: Make AV Seeds kinematically feasible
- Set 1: Modify AV seeds to avoid static obstacles
- Set 2: Modify AV seeds to avoid dynamic & static obstacles

Local Planner Control

The Local Planner Control (LPC) refines the trajectory selected by the ML Ranker. It is an optimization-based algorithm.

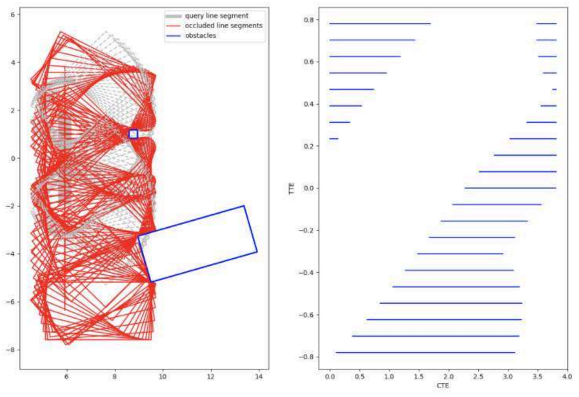
What is the difference between LPC and TREX? TREX provides LPC with a reference trajectory to follow; LPC refines the reference trajectory, optimizing for comfort. LPC runs at a higher rate of execution, about ~50 ms compared to ~200 ms between TREX solutions. This faster execution rate allows LPC to consume fast path predictions, which on occasion drive collision avoidance maneuvers.

The LPC solution is called the LPC Cruise Path, and it is fundamental to the Cruise stack. We use the Cruise path as the input for the lower level controllers.



The image below shows LPC's lateral planning space, where lateral movement solutions are represented.   The left panel represents the planning space in geometric terms, while the right panel represent the same problem in error space. Planning is done in error space.

The red box shows lateral movement.

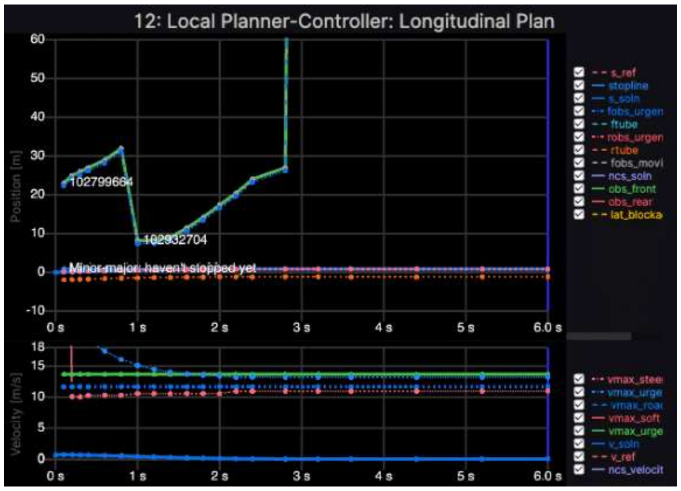


Following the solution of the lateral optimization problem, a solution is also produced in the longitudinal planning space. We have a longitudinal querier to tell us where to stop.

The green line in this Travel vs Time diagram shows where the longitudinal blockages are to guard the longitudinal sweeper files or the trouble profiles for the AV.

The ride bot also gives you a standing of the lateral and the heading difference is allowed for a location.

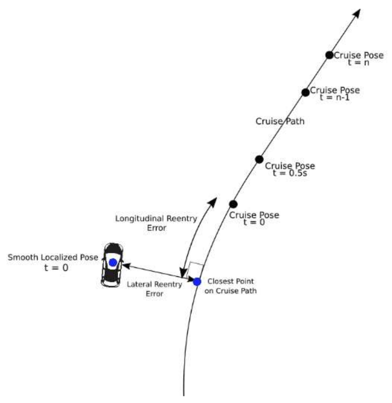
We use a fourth order dynamic basic model to solve the problem. And we use cross functions like reference following, collision avoidance, comfortness, etc.



Vehicle Controls

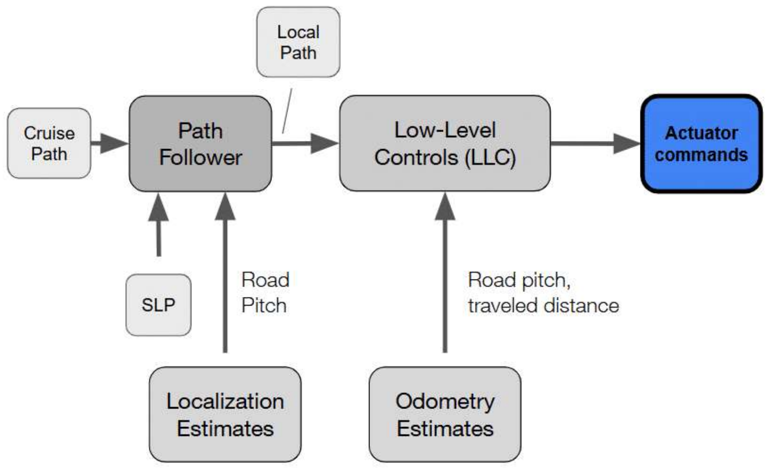
As shown here the AV is grounded to our ground truth at a specific point. Our plan is to follow the planned trajectory, the continuous line below.

If you zoom in you can see that t=0 is where it thinks it is, and our planner thinks that it is at Cruise Pose t=0. There is an error in terms of longitudinal and lateral displacement.



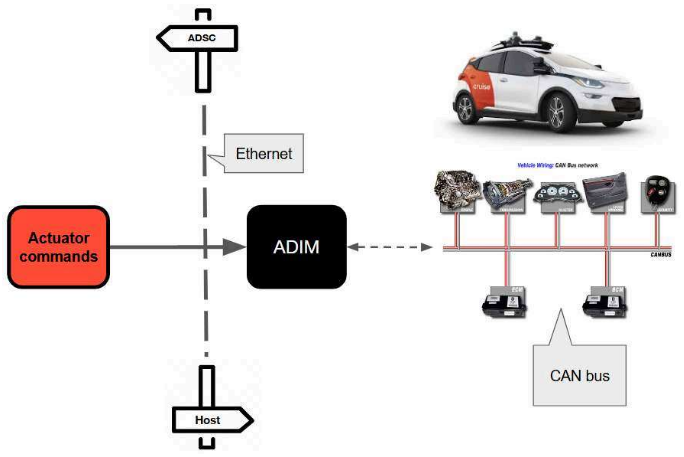
The goal of vehicle controls is to close this gap by taking small actions to decrease this error. This takes into account disturbances like the road pitch. and it provides another version of the planned trajectory (local path) handled by the low-level controls node.

The LLC takes care of some idiosyncrasies of the hardware. It converts acceleration commands into actuator commands (torque, steering angle, etc), considering magnitude and rate limits.



The actuator commands are sent to a module called Advanced Driving Integration Module (ADIM). ADIM converts our commands into CAN Bus packets. Those are accessed by the electronic control units, present in all modern cars. The Bolt has several controllers that are dedicated to various tasks like steering, braking and gear shifting. The level of reliability of these embarked components is much higher than consumer grade computers.

We also receive feedback from the vehicle. For example, we can trigger faults to the stack. ADIM represents our lowest level of safety. If anything goes wrong the ADIM is able to command the car to stop.



Summary

// To be determined.