

Developing the Vehicle-Behavior Interface Layer and Obstacle Field Mood for the 2007 DARPA Urban Challenge



<http://urbanchallenge.case.edu>



Philip Thomas, Department of Electrical Engineering and Computer Science, Case Western Reserve University
Wyatt Newman (Project Mentor), Department of Electrical Engineering and Computer Science, Case Western Reserve University

Vehicle-Behavior Interface Layer (VBIL)

Overview

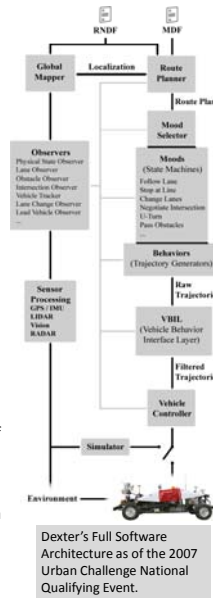
The Vehicle-Controller Interface Layer's original purpose was safety constraint enforcement on the trajectories requested by the higher level artificial intelligence. As such, it enforced two classes of constraints: drivability, and velocity profiling. Upon validating a trajectory, the VBIL also modified it to conform to the input format required by the Vehicle Controller.

Drivability Constraints

If a trajectory is received that Dexter is not capable of driving, it is either rejected or the path truncated, depending on the constraint violated. This can occur when:

- 1) A trajectory contains a turn smaller than Dexter's turning radius
- 2) A trajectory requests a reverse arch when Dexter is not in reverse or is not stopped.
- 3) Dexter is not properly oriented with respect to the trajectory.

In the initial design specification, a violation of any of these constraints would result in the path being completely rejected. As the higher levels of Dexter's AI were implemented, this resulted in deadlocks. Thus, the VBIL was modified to give warnings, and modify requested trajectories to be as close to drivable as possible (small kinks in the path were ignored even though they technically violated the first constraint, if the trajectory had a gear shift while in-motion, Dexter was brought to a full stop just before the shift, etc.).

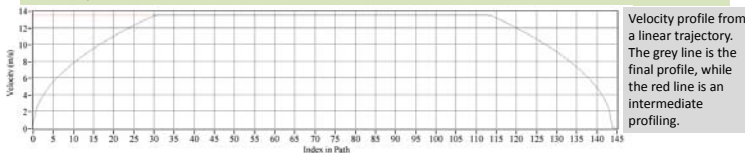


Dexter's Full Software Architecture as of the 2007 Urban Challenge National Qualifying Event.

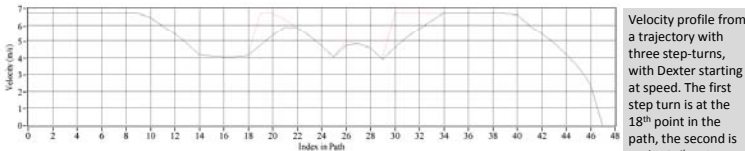
Velocity Constraints

Dexter was never allowed to exceed 13.5 meters per second, so this velocity cap was implemented both in the VBIL, and in the Vehicle Controller. The VBIL also requires that Dexter be able to stop by the end of the trajectory given. This helps to enforce the practice of always being able to stop within the distance known to be safe (assuming trajectories given only traverse areas observed to be obstacle free*).

Once the stop-points have been fixed in the trajectory, and the initial velocity has been replaced with Dexter's current observed velocity, the velocity profile for the entire trajectory is computed as the minimum of the safe profile determined by the VBIL, and the velocity profile given. The VBIL's velocity profile allows for two modes, aggressive and relaxed, which determine the maximum acceleration, deceleration, and centripetal acceleration (limiting the speed around turns). The higher level AI chooses which mode is appropriate for each trajectory. The VBIL was given a short term memory of previous trajectories to avoid feedback issues when profiling velocities starting at Dexter's current speed.



Velocity profile from a linear trajectory. The grey line is the final profile, while the red line is an intermediate profiling.



Velocity profile from a trajectory with three step-turns, with Dexter starting at speed. The first step turn is at the 18th point in the path, the second is at the 25th point, and the third is at the 29th point.

*A module was added to the VBIL by another team member that truncates trajectories that intersect obstacles. Yet another module was added by another team member to maintain proper distance when following a lead vehicle.

Sponsors and Acknowledgements



I would like to thank SOURCE and all the members of Team Case and it's sponsors for giving me this opportunity. None of this would have been possible without the members of Team Case and the massive amounts of software development and testing they all contributed.

I would especially like to thank Wyatt Newman, Amaury Rollin, and Andrew R. Allen for their contributions to the two modules described in this poster. Wyatt Neman was the Team Case Team Leader and my SOURCE Project Mentor, advising me throughout this whole process. Amaury Rollin helped with the initial design of the VBIL, and gave continuous recommendations about how to improve and adapt it as the remainder of the software architecture was implemented above it. Andrew Allen helped with the design and implementation of the Obstacle Field Mood on the LabVIEW side, and supervised and advised me when working within the Mood Selector.

Team Case and the 2007 Urban Challenge

The 2007 DARPA Urban Challenge called for teams to create autonomous vehicles capable of safely navigating an urban environment, handling right of way at stop signs, merging into moving traffic, avoiding in-lane obstacles, and parking. Case Western Reserve University's team, Team Case, consisted of over 60 members, including students, faculty, and industrial collaborators. After over a year of development and testing, Dexter, the team's robot, made it through the Site Visit and Technical Paper reviews to the National Qualifying Event. Dexter placed in the top 20 team of 89 total entrants from around the world, though he did not make it to the finals, to which only 11 teams were admitted.

This endeavor would not have been possible were it not for many generous sponsors. A full list of whom can be found at <http://urbanchallenge.case.edu/sponsors.html>.



Obstacle Field Mood

Overview

The Obstacle Field Mood, also known as the Parking Lot Mood, is the high level AI that uses information on the environment provided by the Observers to generate trajectories through an obstacle field. If directed to park in an obstacle field, Dexter must pill in forwards, park, and reverse out. If he encounters another vehicle, he must tend to the right, giving way to the left.

Implementation

There are two components to the implementation of the Obstacle Field Mood, the first being a planner that constructs an optimized path through an obstacle dense field to a goal configuration, and the second being the mood which causes Dexter to traverse the path, and request new trajectories when new obstacles are observed that interfere with the current plan. The planner was implemented in C++, and the mood was implemented in LabVIEW using the State Diagram Toolkit.

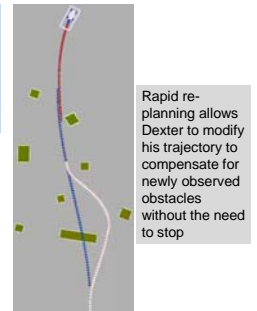
Planner

The planner was a Rapidly-Exploring Random Tree (RRT) that plans a trajectory through the 3-dimensional configuration space consisting of GPS location and heading. RRTs are a sampling based method for quickly determining whether a path exists to the goal state without regard to its optimality. Thus, the algorithm was modified to allow it to grow past the first solution found. After planning for two seconds it terminates and searches the resulting graph for the shortest path to the goal, weighted against reverse arcs. This path is then filtered and returned.

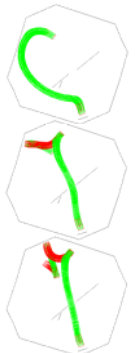
To increase the RRT's efficiency, it was weighted towards the goal, and consisted of trees grown from the start state, goal state, and any chokepoints. Care was taken to allow efficient connecting of all of the trees. To speed up coverage of the configuration space, whenever a new state was found, all states reachable without turning were computed and added to the trees.

The RRT algorithm requires a "simple planner" for connecting nearby configurations. The RRT grew much faster if the heading was computed after the simple planner determined whether the GPS location was reachable with a constant turn radius.

To force Dexter to the right when approaching another moving vehicle head on, moving obstacles were swollen in the configuration space to their front and left side.



Rapid re-planning allows Dexter to modify his trajectory to compensate for newly observed obstacles without the need to stop



The Planner handles chokepoints well, usually finding paths through them even when they are barely larger than Dexter's safety box.

Mood

The mood served as a wrapper for the planner, taking care of initializations (finding the bounding box of the zone, interfacing with the Observers and Global Mapper, etc.), and passing trajectories to the VBIL. It also told the planner whether the goal had to be reached in forwards (for parking), and generated smooth reverse arcs for pulling out of a parking space.

The mood also handled path truncation and re-planning requests for the planner when a new obstacle was observed.

Issues

1. Originally, if Dexter believed an obstacle was inside his safety box (which extends at least one meter on all sides, and two in front), he would refuse to move. When this occurred, to avoid deadlocks, the Obstacle Tracker (Observer) was reset, and Dexter took actions to move away from the obstacle in his safety box.
2. When given a path, Dexter did not always stay within one meter of it, so care had to be taken to force a replan when he strayed too far from his path.
3. The way precision steering was handled meant error in following paths, as calculated in the mood, was compounded whenever Dexter changed direction. This caused frequent replans when the error became greater than one meter.
4. Both the planner and mood had to deal with safety issues derived from noise in obstacle position readings.

For more information, contact Philip Thomas (pst5@case.edu)