Trajectory Planning in a Crossroads for a Fleet of Driverless Vehicles

Olivier Mehani[1] and Arnaud de La Fortelle[1,2]

[1] Inria Rocquencourt, Imara Team, Domaine de Voluceau, BP 105, 78153 Le Chesnay Cedex, France, olivier.mehani@inria.fr
[2] École des Mines de Paris, CAOR Research Centre, 60 boulevard Saint-Michel, 75272 Paris Cedex 06, France, arnaud.delafortelle@inria.fr

1 Framework and Problem Setting

In a transportation system with fully autonomous vehicles (or cybercars), a lot of tasks have to be executed to answer the demand of a customer to travel between two points. The framework of this paper is an approach consisting in decomposing the planning into three levels, each of which using only a relevant subset of the information, thus reducing the complexity:

the macroscopic level, *e.g.* a city or a region;
the mesoscopic level, *e.g.* a city quarter;
the microscopic level, *i.e.* the surroundings of the cybercar.

At the macroscopic level, a *path* is computed. By path, we mean a succession of edges (road segments) in the graph description of the road network.

At the mesoscopic level, paths are transmitted by the upper level and turned into *trajectories*. A trajectory is a precise space-time curve that the cybercar has to follow. Typical precisions are 10 cm and 1/10 s. The goal of this level is to produce trajectories for all the cybercars circulating in a given area. These trajectories have to be safe, most notably collision-free, but also efficient, *i.e.* deadlock-free.

At the microscopic level, the cybercar's control moves along the trajectory and ensures that none of these collisions which couldn't have been foreseen at the higher levels occur.

It should also be stressed that this framework is not top-down only. Each lower level can raise warnings to higher levels ; *e.g.* stopping before an unforeseen children raises a warning from the microscopic level, where it is detected, to the mesoscopic level, where new trajectories are calculated.

In this paper, we focus on the mesoscopic level. More precisely, we present the study on a simple crossroads (X junction) of the concepts developed to deal in a general context. Several trajectory-generating algorithms are described and compared in terms of efficiency. One key element of these algorithms is the *respect of the controllability constraints of the vehicles*, for the actual vehicles to be able to follow the generated trajectories. A second key element is the *management of the shared resources*, *i.e.* the places on the junction where trajectories intersect.

2 Model and Simulation

On first approximation, several simplifying conditions are assumed: all cybercars are *identical*, the server attributing trajectories knows the state of all relevant cybercars with no delay, *i.e. perfect communication*, and the microscopic level is perfect, *i.e.* cybercars *exactly follow the trajectories* attributed by the server.

The crossroads being fixed, the paths from every entry point to each exit point can be precalculated. This results in a set of 12 (in our case) fixed two-dimensional traces, compatible with the vehicles physical constraints (continuously changing wheels angle,...) ; curves generated using clothoids have been found to have these properties, hence they are used to construct these traces. Thus, the task of the algorithms studied thereafter is to compute the speed at which each vehicle should follow its trace in order not to collide or block any other vehicle.

In order to evaluate these algorithms, a simple simulator has been written. It simulates a regular crossroads, with two 2-ways roads crossing at a right angle. The introduction law of the vehicles can be tuned, and each vehicle is given an itinerary composed of an entry and an exit directions out of the four possible ones (North, South, East, West): this is the information allegedly transmitted by the macroscopic level.

The first simplistic algorithm is a polling algorithm, allowing only one vehicle to enter the crossroads at a time. This obviously guarantees the collision-safety of the crossroads, at the price of a very low throughput of the crossroads and high waiting times for the vehicles.

A much more efficient algorithm is a modification of a reservation-based intersection control mechanism [1,2], modified to consider those points of the path that are potentially dangerous rather than all the crossroads space. Given all the possible traces on the crossroads, the set of *critical points* where they intersect is computed. When entering the crossroads, a vehicle evaluates the time it will take, given its current speed, to reach each critical point, and how long it will take to pass it. This information is used to build a reservation request, which is accepted or rejected by the infrastructure. If the reservation is accepted, the vehicle tries to accelerate and to place a new reservation. If it has been rejected, the vehicle must stop before the first critical point of its trajectory, or before the last vehicle already stopped at said critical point.

References

1. Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 530–537, New York, New York, USA, July 2004.
2. Kurt Dresner and Peter Stone. Multiagent traffic management: An improved intersection control mechanism. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 471–477, Utrecht, The Netherlands, July 2005.