A Lightweight Simulator for Autonomous Driving Motion Planning Development

Tianyu Gu
Electrical & Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213, USA
Email: tianyu@cmu.edu

John M. Dolan
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
Email: jmd@cs.cmu.edu

***Abstract*—A good simulation environment will facilitate motion planning algorithm development for urban autonomous driving. The first requirement of such a simulator is to be able to replicate a complex urban environment, including road network, curb, general objects, etc. The second requirement is to simulate a realistic host vehicle, which includes perception, control and vehicle dynamics, to recreate imperfect inputs and non-accurate execution of the planner. The third requirement is to model traffic participants (other on-road vehicles) for microscopic traffic simulation. Intelligent-agent-based techniques are used to allow the traffic participants to interact with the environment and each other. In this paper, we present an open-source lightweight simulation environment, FastSim, which is designed to meet the three requirements above.**

Keywords–Traffic simulator; Intelligent agent; Motion Planning; Autonomous driving

I. Introduction

Autonomous passenger vehicles (APV) have demonstrated promising social impacts that touch nearly all aspects of modern transportation. Motion planning (MP) algorithms are one of the most important components in such autonomous systems. The development of a planning algorithm is typically first performed in a simulator before applying it on the actual robot for convenience and safety reasons. Many simulation environments are either too complex or overly-simplified [1][2][3], and only a few are freely distributed for community usage. In this paper, we present the development of a lightweight and real-time simulation environment designed specifically for quick motion planning algorithm prototyping in urban environments, where the host vehicle operates on roads or freeways with structured lane information.

A. Related Work

MP algorithms take inputs from the upper-level perception processing module and generate outputs to the lower-level controller modules. From the planner's perspective, three factors must be reproduced in a simulator: the ground-truth of the surrounding environment, the perception of the ground-truth, and the host vehicle dynamics where the actual execution result of the planned actions is evaluated.

In terms of the environment ground-truth, Carnegie Mellon University Grand Challenge team [2] proposed a simulation package for desert vehicles with general object representations for field navigation, but no capability to model common objects in urban environments. The Tartan Racing Urban Challenge System (TRUCS) [3] explicitly represented different types of moving objects in urban environments, but was not capable of modeling the interactive capabilities of many objects. For a simulation environment that aimed at creating realistic microscopic traffic, [1] proposed a lane changing and merging model for on-road vehicles. However, these models used simplified assumptions, which can only react to other in-lane vehicles, but not to other environment objects like static objects and pedestrians, etc.

Self-localization and sensing the surrounding environment are the two pillars of perception. Prior simulators typically assume perfect localization. In order to reproduce realistic imperfect localization, a closer look at actual localization methods used in reality and the nature of output dynamics is required. For environment sensing, the majority of prior simulation environments directly feed the motion planning algorithms with complete knowledge (directly pass the simulated ground-truth). However, on real robot, sensing is never perfect, e.g., [4] investigated the real-world perception failure cases with real Light Detection and Ranging(LIDAR)-based ranger. For MP to behave robustly on robot, imperfect sensing (e.g., sensor limitations) is important to simulate for MP development.

In terms of the host vehicle dynamics, there is a huge body of literature in vehicle modeling [5]. Many prior simulators used an overly simplified vehicle model. On the other hand, too-complicated vehicle models would be computationally unjustifiable. Meanwhile, lower-level vehicle controllers (e.g., path tracking and speed regulation) are external to the motion planner, hence must also be simulated, preferably with the actual controllers [6] implemented on the robot itself.

In the remainder of this paper, we explain the design of the proposed real-time simulation environment FastSim for MP algorithm development. The organization of this paper is as follows. Section II explains the implementation details of the proposed simulation environment FastSim. Section III presents the computationally efficient implementation of intelligent-agent-based microscopic traffic simulation. Section IV describes the user interface design. Section V concludes with our contributions and future work.

II. Simulation Environment

Based on the requirements from section I-A, the FastSim simulator consists of three simulation engines (Figure 1):

- The environment simulation engine models different invariant (e.g., road network, curb, etc.) and varying world elements (e.g., general static or moving objects).
- The perception simulation engine models the imperfect self-localization and environment sensing.
- The host vehicle simulation engine models the vehicle dynamics and low-level tracking controllers.

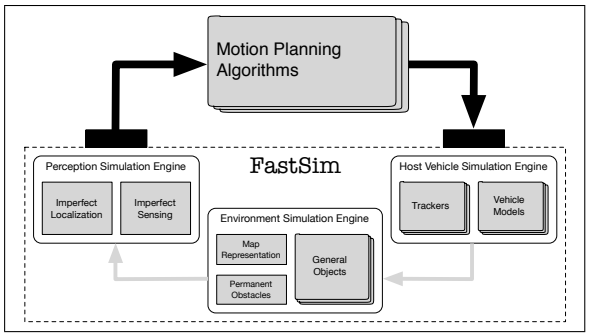

Figure 1. System diagram of FastSim.

A. Environment Simulation

1) Road Network: A road network provides the interconnectivity of roads and roads' lane-level information that specifies the drivable regions. The route network definition file (RNDF) [3] is a robust way to store road network information. It makes use of road segments, where each segment contains one or more parallel lanes. Each lane is specified by a series of global way-points. Connectivity among lanes is defined by pairs of exit/entry way-points. One main limitation of the RNDF is its restriction to a fixed lane width and speed limit for each lane.

FastSim uses a similar waypoint-based lane definition. But for each waypoint, lane width (w) and speed limit (v_{lim}) are added to global position (x and y). An ever-increasing station coordinate (s) is first calculated for each waypoint by calculating piecewise-linear cumulative distance along-road. Cubic polynomial or step signal could be used for smooth or immediate interpolation:

$$\begin{cases} X(s) = \sum_{i=0}^{3} p_X^i \cdot s^i \\ X(s) = X^i|_{s^i \le s \le s^{i+1}} \end{cases} \quad (1)$$

when X is global position (x, y), it is commonly interpolated by cubic polynomials. The first-order (heading θ) and second-order (curvature κ) geometric information is also easily obtained via analytic differentiation. When X is lane width (w) or speed limit (v_{lim}), either interpolation could be used according to the specific situation.

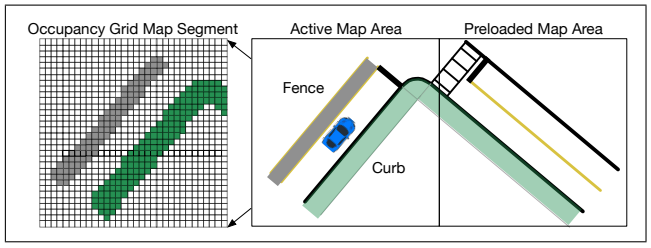

Figure 2. Permanent obstacles.

2) Permanent Obstacles: Permanent obstacles (Figure 2) refer to the stationary environment constraints that make certain regions non-traversable, such as curb and lane fences. Unlike general objects below, permanent obstacles typically do not have a separable shape. Hence, we use an occupancy grid representation as an off-line map file, which is further rasterized at a larger scale to break a large area into smaller pieces. As the host vehicle moves, segments of occupancy grids maps in proximity to the vehicle are loaded.

3) General Objects: Various static and moving objects must be modeled in the urban environment. Objects of different types have different motion dynamics. The trivial non-movement model is for static objects (e.g., trash bins), which only contains unchanging pose information; a particle movement model is used for objects whose motion can be omni-directional (e.g., pedestrians); the kinematic bicycle model can be used to model objects with non-holonomic kinematic constraints (e.g., bicyclists and other passenger vehicles).

As for the motion of objects, it is sometimes useful for other objects to follow a predetermined trajectory and not react to other simulated objects, for objects like the leading car in a queue of traffic, or a reckless pedestrian crossing the street disregarding traffic. In other cases, it is more important to create more realistic on-road traffic by enabling traffic participants with some intelligence to interact. Section III will explain more details on this matter.

B. Perception Simulation

In a realistic robot system, both localization and environment sensing have errors and limitations. One of the main design goals of the simulator is for it to be sophisticated enough to model such imperfect conditions for MP algorithm design purposes. The environment simulation engine above provides the "ground-truth", hence we need a separate perception simulation module to mimic realistic perception outcomes.

1) Imperfect Localization: The majority of localization methods are based on Extended-Kalman-Filter (EKF), Monte-Carlo-Filter (MCF) or Simultaneous-Localization-and-Mapping (SLAM) algorithms. They output best estimates of the vehicle state, along with covariance matrices describing the confidence of measurement. However, localization error in reality is largely situation-dependent, e.g., the vehicle loses GPS in an urban canyon or enters an area where the environment's features are quite different from the map. It is extremely difficult to model these realistic scenarios in a simulation environment. However, to make sure the perception outputs to planners are compatible with that of a real perception system, we add arbitrarily biased white-noise to the ground truth, and apply an EKF to maintain a filter-based perception output.

2) Imperfect Object Sensing: Ranger-based sensing units are the most commonly used on an APV. Two main sources of imperfect (partial) perception are the limited field of view (FOV) and occlusion. We simulate these limitations by the placement of virtual sensors at different configurations on the host vehicle (Figure 3). At each time-stamp, a constant-horizon line-tracing algorithm is used to simulate the sensor scanning, and only the objects that are reached by the simulated detection rays are made visible to the MP algorithm.

C. Host Vehicle Simulation

In a realistic robot system, the plan is never executed perfectly due to actuation errors and unmodeled vehicle dynamics. It is important for FastSim to simulate the execution of motion plans with adequately sophisticated host vehicle models for MP algorithm evaluation purposes.

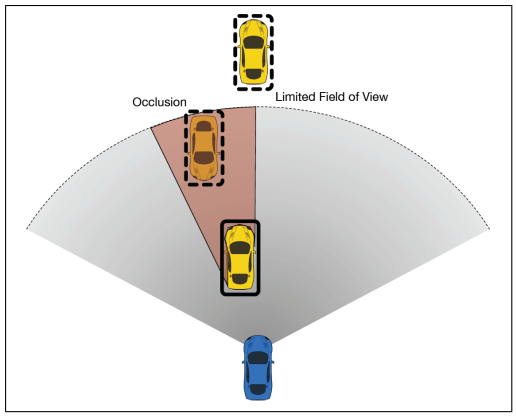

Figure 3. Imperfect sensing due to limited field of view and occlusion.

1) Tracking Control: In most APV systems, a decoupled planning and control scheme is used: the output of MP algorithms is fed to lower-level tracking controller and execution components. From the motion planner's perspective, the controller contributes partially to the overall vehicle dynamics. It is hence necessary to model the controllers. Two commonly used tracking controllers, a pure-pursuit and Linear Quadratic Regulator (LQR)-based trajectory tracker [6], are implemented.

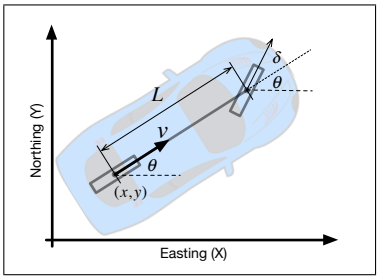

Figure 4. Dynamic bicycle model.

2) Vehicle Dynamics: A simplistic kinematic model has been used for many ground robot simulations [3]. While suitable for low-speed navigation applications, they cannot model realistic high-speed vehicle dynamics. Two of the most important factors are latency and vehicle skidding dynamics. Hence, we adopt a dynamic bicycle model (Figure 4):

$$\begin{cases} \dot{x} = v \cdot cos(\theta) \\ \dot{y} = v \cdot sin(\theta) \\ \dot{\theta} = G_s \cdot \dfrac{v}{L} \cdot tan\delta \\ \dot{\delta} = \dfrac{1}{T_\delta} \cdot (\boldsymbol{\delta_c(t)} - \delta) \\ \dot{v} = a \\ \dot{a} = \dfrac{1}{T_a} \cdot (\boldsymbol{a_c(t)} - a) \end{cases} \tag{2}$$

where $G_s \in [0, 1]$ is the slipping coefficient, T_δ and T_a are actuation latency coefficients of first-order low-pass filters, x, y and θ are the global pose, v is the speed scaler, δ and a are the actual steering/acceleration scalars, and δ_c and a_c are the commanded steering/acceleration (model inputs).

III. Modeling Traffic with Intelligent-Agents

As explained in Section II-A3, it is sometimes important to simulate basic interactive intelligence of other traffic participants in order to recreate realistic traffic behavior. In this paper, we are primarily concerned with surrounding on-road vehicles, particularly, interested in modeling three basic maneuver capabilities:

- $\mathcal{M}_1$: Swerve avoidance of static obstacles.
- $\mathcal{M}_2$: Longitudinal avoidance of/distance keeping to a leading object.
- $\mathcal{M}_3$: Lane-changing maneuver.

The challenge is to model these behaviors in a computationally efficient manner. It is natural to think of using a planner-based approach for each simulated on-road vehicle. However, this is generally not scalable if the number of on-road vehicles is large. In this section, we propose a computationally efficient interaction model for other on-road vehicles capable of performing the three maneuvers above.

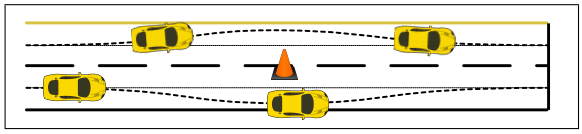

Figure 5. $\mathcal{M}_1$: Swerve avoidance of static obstacles

For $\mathcal{M}_1$, the key is to plan a vehicle-independent reference trajectory once per lane per cycle, and reuse this plan for multiple on-road vehicles. We make use of the elastic-band algorithm [7] to generate one reference trajectory per lane of interest per cycle, so that all the moving objects in that lane can reuse this planned trajectory (Figure 5).

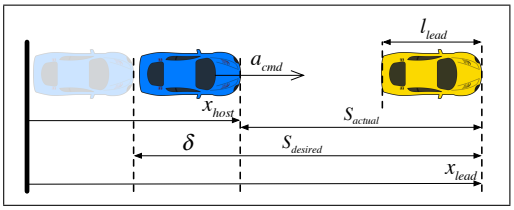

Figure 6. $\mathcal{M}_2$: Longitudinal avoidance /distance keeping to a leading object

Then the individual vehicles only need to implement cheap-to-evaluate lateral and longitudinal controllers to track the planned reference. For lateral control, tracking controllers described in section II-C1 could be reused. For longitudinal control ($\mathcal{M}_2$), a constant-time adaptive cruise controller [8] is implemented to perform distance keeping and slowing-down based on the relative distance to the leading object (Figure 6):

$$\begin{cases} S_{desired} & = l_{lead} + T \cdot \dot{x}_{host} \\ S_{actual} & = x_{lead} - x_{host} \\ \delta & = S_{desired} - S_{actual} \\ a_{cmd} & = -\frac{1}{h} \cdot (-\dot{S}_{actual} + \lambda \cdot \delta) \end{cases} \tag{3}$$

where $S_{desired}$ and S_{actual} are the desired and actual longitudinal gaps between two vehicles, δ is the difference between these two gaps, l_{lead} is the length of the leading vehicle, x_{host} and x_{lead} are the longitudinal positions of host and leading

vehicle, T is the time coefficient of the controller, h and λ are the tunable coefficients to modify the aggressiveness of the controller, and a_{cmd} is commanded acceleration, which is the output of the controller.

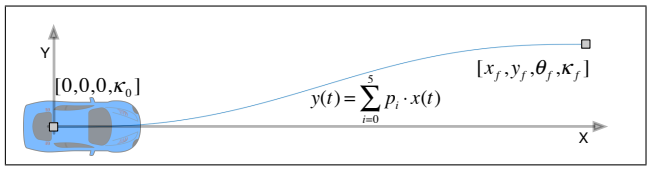

Figure 7. $\mathcal{M}_3$: Lane-changing maneuver.

For $\mathcal{M}_3$, a single lane-change path is generated using a polynomial [9] that connects from the current state of the object to a look-ahead state in the target lane (Figure 7):

$$y(t) = \sum_{i=0}^{5} p_i \cdot x(t) \tag{4}$$

where the polynomial coefficients p_i can be found analytically, hence computationally trivially. The look-ahead distance is an empirical function of its current speed.

Depending on the nature of traffic, lane change can be free or cooperative. The former is trivial. For the latter, if the same controller for $\mathcal{M}_2$ is used, a significant change in the spacing (after lane-change) between two simulated vehicles will cause huge deceleration and generate a slowing-down shock-wave effect on all following vehicles in the lane. We adopt the controller proposed in [1] to simulate smoother cooperative lane change in dense traffic. Refer to the original paper for more details.

IV. Results

The proposed simulator has been used to develop urban driving motion planning algorithms [7] for the autonomous Cadillac SRX testbed [10]. Compared with the simulator used in the 2007 DARPA Urban Challenge [3], "FastSim" is capable of modeling host vehicle with more accurate dynamics by using dynamic bicycle model, so that the vehicle response at higher-speed can be replicated. Meanwhile, by directly modeling perception system, "FastSim" is capable to recreate the non-perfect sensing limitation imposed by realistic sensors to create challenging test cases for the motion planner. Finally, "FastSim" can model various dynamic objects with more flexible motion patterns, and traffic pattern which is important for simulating urban driving scenarios.

A graphic user interface for FastSim is implemented to facilitate real-time monitoring and manipulation (Figure 8). It consists of six main functional components: world plotter (A), XML-based scenario loading area (B), historical host vehicle measurement (C), world plotter zoom/panning tool (D), simulator/planner stop/go toggle tool (E) and external trigger control panel (F). More description and example usage of FastSim can be found in [11].

V. Conclusion

In this paper, we propose a lightweight simulation environment FastSim for rapid MP algorithm development for urban autonomous driving. Three cornerstone simulation components, i.e., surrounding environment, perception and host vehicle control/execution, are modeled to create a sufficiently complex environment for MP algorithm evaluation. We further proposed efficient models to reproduce basic interaction intelligence of other on-road traffic participants to create a more realistic simulation environment. FastSim is designed with modular programming, hence the models of different simulation components can be swapped for different research projects easily.

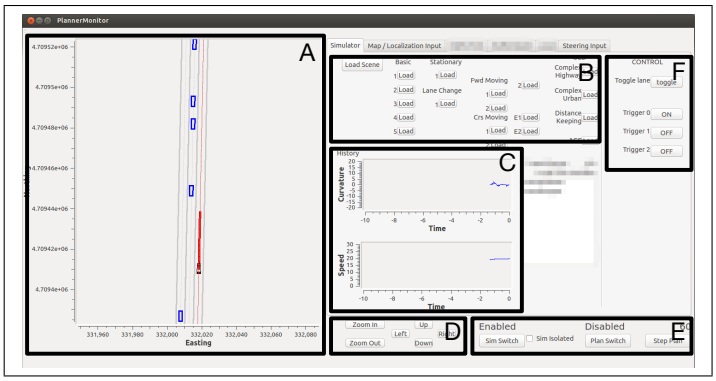

Figure 8. The graphic user interface of FastSim.

In the future, the simulation environment will be further expanded to be compatible with unstructured environments like a parking lot. More moving object models will also be implemented, such as the trailer model for trucks. In addition, more interaction intelligence of other non-vehicle traffic participants like pedestrians and bicyclists will also be investigated and modeled.

References

[1] P. Hidas, "Modelling vehicle interactions in microscopic simulation of merging and weaving," Transportation Research Part C: Emerging Technologies, vol. 13, no. 1, 2005, pp. 37–62.

[2] C. Urmson et al., "High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004," Robotics Institute, CMU, Tech. Rep. CMU-RI-TR-04-37, 2004.

[3] M. McNaughton et al., "Software infrastructure for an autonomous ground vehicle," Journal of Aerospace Computing, Information, and Communication, vol. 5, no. 12, 2008, pp. 491–505.

[4] R. MacLachlan, "Tracking moving objects from a moving vehicle using a laser scanner," Robotics Institute, CMU, Tech. Rep. CMU-RI-TR-05-07, 2005.

[5] W. Milliken and D. L. Milliken, Race car vehicle dynamics. Society of Automotive Engineers Warrendale, 1995, vol. 400.

[6] J. M. Snider, "Automatic steering methods for autonomous automobile path tracking," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08, 2009.

[7] T. Gu et al., "Tunable and stable real-time trajectory planning for urban autonomous driving," in IEEE/RSJ International Conference on intelligent Robots and Systems (IROS 2015). To appear.

[8] R. Rajamani, Vehicle dynamics and control. Springer Science Business Media, 2011.

[9] J.-W. Lee and B. Litkouhi, "A unified framework of the automated lane centering/changing control for motion smoothness adaptation," in Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on. IEEE, Conference Proceedings, pp. 282–287.

[10] J. Wei et al., "Towards a viable autonomous driving research platform," in Intelligent Vehicles Symposium (IV), 2013 IEEE. IEEE, Conference Proceedings, pp. 763–770.

[11] T. Gu and J. Dolan, "Github page for FastSim"." [Online]. Available: http://www.tianyugu.net/publications.html