Motion Planning for Autonomous Vehicle Parking

Ashish Roongta and Prateek Parmeshwar

***Abstract*— This paper presents an architecture for simulation and high-fidelity testing for motion planning of autonomous vehicles in unstructured environments such as parking lots. A lattice based planner was developed for optimal and efficient plans for parking the ego vehicles. The computed vehicle trajectory is tracked with the help of a LQR based lateral and a PID longitudinal controller. Carla simulator was used for this project to allow a realistic physics-enabled scenario generation. This architecture can be used for motion planning for realistic parking of autonomous vehicles simulating congestion avoidance, crashes, conflict, back and forth negotiation.**

I. Introduction

Americans spend an average of 17 hours per year searching for parking, resulting in a cost of $345 per driver in wasted time, fuel and emissions[1]. Over 50,000 crashes occur in parking lots and parking garages annually, resulting in 500 or more deaths and more than 60,000 injuries. While on-road autonomous driving technologies are still years away from successful deployment, park assist technologies have already been deployed with the current stack of vehicles. These technologies are still insufficient in terms of complete autonomous functionality and efficiency in large crowded parking spaces. Despite seeming very promising, smart parking technologies such as the Tesla Summon have recorded a very high disengagement and failure rate.

We intend to develop an architecture to support rapid development and high fidelity testing of planning for autonomous vehicles in unstructured environments. This paper presents the architecture for a lattice based global planner, local planner for trajectory optimization, dynamics based vehicle controller, and physics-enable realistic simulation of complex parking scenarios. Currently, the motion planning has been developed for a single vehicle, with potential to be scaled for multiple vehicles.

The final aim of this project is to develop a motion planner for parking of multiple autonomous vehicles in a busy parking lot. We intend to analyze and compare autonomous vehicle paradigms with synchronized planning (connected vehicles - shared states and goals between all vehicles) and unsynchronized planning (no shared information). This would lead to a quantitative estimation of possible gain in efficiency with connected vehicles. Currently, the architecture involves motion planning and control for a single ego vehicle.

II. Approach

The entire architecture consists of three subsystems: the motion planner, vehicle controller and the simulator.

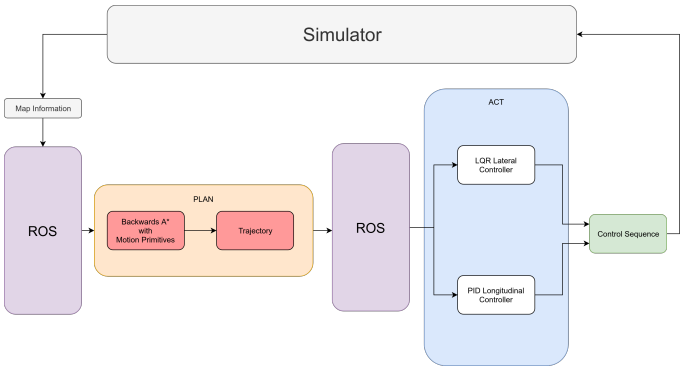

Fig. 1. System Architecture

Carla, a high fidelity open-source driving simulator was used for simulation. A large parking lot with over 110 parking spots was developed for this project. For simplicity of architecture, each parking spot was designed to be identical in dimensions with sufficient open space between consecutive rows of parking spots.
The simulator (Carla) subsystem provides the map information: parking spot locations, availability of parking spots, and instantaneous states of all the vehicles in the environment. This information is published on a ROS topic which the planning subsystem subscribes to. A static occupancy grid is generated by the planning subsystem, based on the map and vehicle state information. The computed optimal trajectory generated for each vehicle via the motion planning algorithm is communicated to the controller subsystem. The controller outputs the required throttle, brake and steering commands in order to track the reference trajectory and communicates with the Carla Server. ROS was used mainly as a communication protocol between the motion planning subsystem in C++ and the vehicle controller in Python.

A. Planning Representation

The global planner runs a graph search where each node in the graph is given by the vehicle's state. A state in the global plan includes the following,

$$State = [x\ y\ \theta]$$

The actions that can be taken at any time-step are based on motion primitives that are derived from the vehicle's

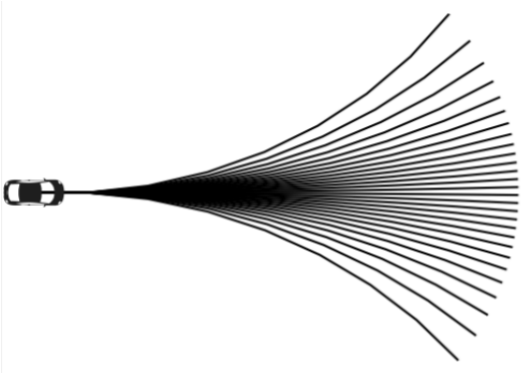

Fig. 2. Forwards motion primitives

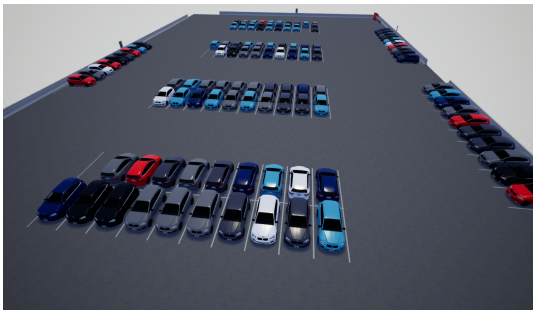

Fig. 3. Parking lot developed for the simulator

kinematic constraints. Every motion primitive was assigned a cost in terms of the difficulty of the maneuver.

Assuming that the vehicle moves with a constant speed, the state update was determined as follows,

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \omega \tag{1}$$

Given the rate of chage of pose, the pose at the next time step was determined by Euler integration[6].

$$\begin{aligned} x_{t+dt} &= x_t + v * \cos(\theta) * dt \\ y_{t+dt} &= y_t + v * \sin(\theta) * dt \\ \theta_{t+dt} &= \theta_t + (v \backslash L) * \tan(\varphi) * dt \end{aligned} \tag{2}$$

where v is the vehicle speed, L is the length of the vehicle and φ is the steering angle.

B. Planning algorithm

After obtaining map information from the simulator, the planning subsystem generates an occupancy (obstacle) grid at a predefined discretization. A backwards weighted A^* search in the state defined using motion primitives was then run to obtain a global plan from the start to the goal. A backwards search was chosen as the goal location (parking spot) is typically more restricted than the start location. Given the choice of algorithm used, completeness is guaranteed but the solutions might be sub-optimal. A local planner for trajectory optimization was then run between

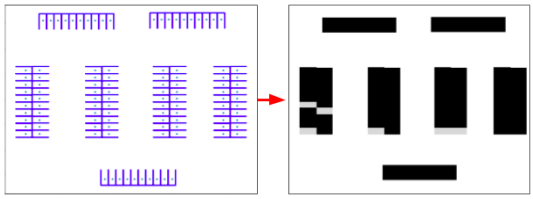

Fig. 4. Parking lot map and Occupancy grid

two nodes of the global plan to ensure continuity between the nodes of the global and that the controller is able to track this reference trajectory closely. The local planner itself consisted of minimum jerk trajectories generated by solving a boundary value problem between two nodes. A 2D Dijkstra's search was used as a heuristic to make search faster. Other optimizations included precomputing actions and vehicle footprint for collision checking at every pose of the vehicle. Both of these were calculated at an initial pose and later transformed using a homogeneous transformation matrix to generate new motion primitives (set of actions) at a new pose.

For collision checking, the ego-vehicle was modeled as two circles. This representation ensured that the computational overhead for collision is not too high but also that the collision check itself is accurate to a large extent.

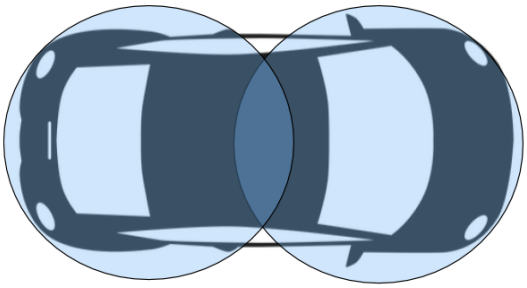

Fig. 5. Ego-vehicle represented as two circles for collision check

The reference trajectory generated by the planner encoded x, y and v i.e. the position and the longitudinal velocity. This reference trajectory was then published for the controller to utilize. The controller has two modules which are a PID longitudinal controller and a LQR lateral controller.

III. Experiments and Analysis

Initially the action space included motion primitives both in the forward and the backward direction. The backward direction motion primitives warranted additional optimization and velocity optimization at switch points, as well as an additional controller design for the backward direction. Due to the limited time scope for this study, we restricted the action space to motion primitives in the forward direction only.

The motion primitives were weighted proportionally to the change in the steering angle, i.e. difficulty of the maneuver. This resulted in generation of feasible (straight) reference

trajectories by the motion planning algorithm.

The first choice for heuristics was a 2D A* search on the

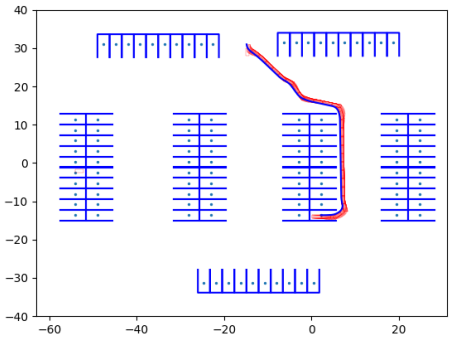

Fig. 6. Reference trajectory generated by Backward A* search

occupancy grid, to allow efficient online heuristic computation. The overhead of A* at each planning iteration and state expansion was still too high, hence we switched to a pre-computed 2D Dijkstra's search. The choice of backward A* search for the 3-dimensional problem guaranteed a straight, collision-free entry to the parking spots for all the ego vehicles.

The following were pre-computed for efficient and fast planning:

1) Motion Primitives
2) 2D xy Dijkstra's heuristic
3) Vehicle footprint along the motion primitives

The results can be viewed in the videos (hyperlinked) below:

1) Motion Planning for AV I.
2) Motion Planning for AV II.

NOTE: Following are the links to the videos if the hyperlink does not work:

1) https://youtu.be/d6bgpWKP3xQ
2) https://youtu.be/6QaFjEKIHG0

IV. CONCLUSIONS

The architecture developed through this work can be used for realistic testing and simulation of motion planning for vehicle in unstructured environments (parking lots). Planning for parking in tight environments requires extremely accurate representation of the environment and an efficient motion planning approach. Collision checking is the most computationally expensive fragment of motion planning in tight spaces. Utmost use of pre-computation of different attributes of motion planning is necessary to allow efficient and fast generation of reference trajectories in complex crowded scenarios.

V. FUTURE WORK

The architecture developed currently successfully simulates planning and control of a single ego vehicle in tight parking lots. We intend to use the architecture for a detailed analysis on motion planning of multiple vehicles in the synchronous and and asynchronous paradigms in the future. The vehicle controller and simulation architecture developed during this study can already be used for control and simulation of multiple vehicles. Incorporating the time attribute, scaling the motion planning to multiple vehicles and further updating to an anytime D* planning algorithm would allow simulation of complicated parking scenarios involving traffic congestion and negotiation. Further, the use of Carla simulator would allow an easy communication and sharing of vehicle states on-demand.

ACKNOWLEDGMENT

We would like to thank Dr. Maxim Likhachev for exceptional lectures on the course:16782:Planning and decision making in Robotics, and technical advice throughout the course of this study.

REFERENCES

[1] http://inrix.com/press-releases/parking-pain-us/
[2] Dave Ferguson, Thomas M. Howard, Maxim Likhachev *Motion Planning in Urban Environments: Part I*
[3] Dave Ferguson, Thomas M. Howard, Maxim Likhachev *Motion Planning in Urban Environments: Part II*
[4] Zlatan Ajanovic, Bakir Lacevic, Barys Shyrokau, Michael Stolz, Martin Horn *Search-based optimal motion planning for automated driving*
[5] Bai Lia Zhijiang Shao *A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles*
[6] https://github.com/architkhullar/RobotMotionPlanning_TermProject/