SPEIT SJTU

ICT Project1 Intelligent Vehicles - Mini Project

Report on 'Local Path Planning for Off-Road Autonomous Driving With Avoidance of Static Obstacles'

Tiance WU
117260910057

January 3, 2020

上海交大-巴黎高科卓越工程师学院
Ecole d'Ingénieurs SJTU-ParisTech

1 Introduction

The paper ‘Local Path Planning for Off-Road Autonomous Driving With Avoidance of Static Obstacles’ is written by Keonyup Chu, Minchae Lee and Myoungho Sunwoo on IEEE Transactions on Intelligent Transportation Systems in 2012. In this paper, the authors proposes an automatic path planning method specially designed for vehicles moving on the road. While being driven automatically on the road, vehicles could encounter different types of obstacles: some of them are fixed, some of them are mobile or uncertain. As it is nearly impossible to predict all obstacles and set up a plan of the whole route in the beginning, vehicles have to detect obstacles while moving and choose a most suitable path according to the objects detected. The system proposed in this paper tends to solve this problem.

Generally, the system that the authors proposed is a complete combination of different traditional methods: baseline construction via cubic splines, localization, transformation from Cartesian Coordinates to Curvlinear Coordinates, Collision Risk prediction via Gaussian Convolution, path selection via predicted smoothness and risk factors, etc. Nearly every part of the system is individually not new, but their combination shows good performance on path planning considering obstacles. The authors also implemented their system on a Hyundai experimental autonomous vehicle and got positive experimental result. The report of this system will be seperated in three parts. In Section 2, the structure of the system and the theoretical methods used for each part of the system will be briefly introduced. In Section 3, a brief implementation of the proposed system and its result is presented. In Section 4, we will conclude our report.

2 Method in Theory

Figure. 1 shows a system overview. The whole process consists of four steps: creation of baseline of route via Cube Spline Curve Fitting, transformation from Cartesian Coordinate(x,y) to Curvilinear Coordinates(s,v), localization of the vehicles on the Coordinates, path generation via curve fitting, and path selection considering obstacles, smoothness of the path as well as the speed of the vehicle.

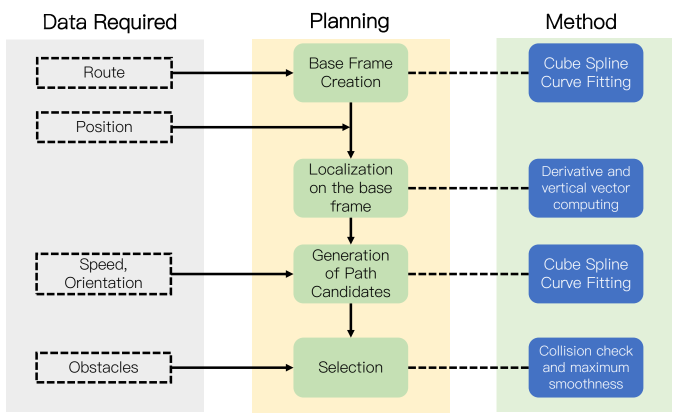
Data Required
Planning
Method
Route
Position
Speed, Orientation
Obstacles
Base Frame Creation
Localization on the base frame
Generation of Path Candidates
Selection
Cube Spline Curve Fitting
Derivative and vertical vector computing
Cube Spline Curve Fitting
Collision check and maximum smoothness

Figure 1: System Overview

2.1 Baseline Generation

The authors choose **Cubic Spline Curve Fitting** for baseline generation. All paths chosen for vehicles has to be exactly on the route(road). For a route on a 2-D plan (x,y), each horizontal value x might have several different corresponding vertical values y, and each y value could also have multiple corresponding x values. Creating a function *y=f(x)* to describe the route is therefore unrealizable. However, if we consider the distance s from the beginning of the route to the current position, we could see each s value could correspond a single (x, y) tuple, whenever a route do not goes back. Therefore it is required to generate two functions $x = f_1(s)$ and $y = f_2(s)$.

As referred by the author, the formation of the baseline requires highly on smoothness for 2-grade derivatives. Therefore, it is supposed to use cube functions for f_1 and f_2 for a cube function supports smoothness on 2-grade derivatives. Experimental result also shows that cube spline curve fitting has good performance on baseline generation. If we have several waypoints already detected, $\{s_1, s_2, ..., s_n\}$, the relation between x, y and s could be described as

$$\begin{aligned} x &= a_1(s - s_i)^3 + b_1(s - s_i)^2 + c_1(s - s_i) + d_1; \\ y &= a_2(s - s_i)^3 + b_2(s - s_i)^2 + c_2(s - s_i) + d_2; \end{aligned} \tag{1}$$

There are four variables for x and four variables for y. Simply, if we have 4 waypoints, we could simply resolve all variables. If we have more than 4 waypoints, we seperate then on different groups of 4 points and generate baselines for them. Figure. 2 shows a baseline created with several way points.

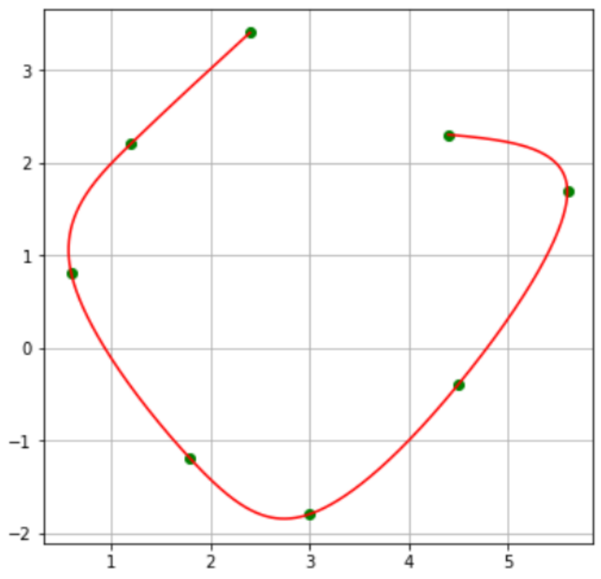

Figure 2: Baseline creation on plan(x, y). Waypoints are green. Baseline is red.

2.2 Coordinate Transform

This section shows how to transform the coordinate(x, y) to a curvilinear coordinate(s, v). For (s, v), s represents distance value on the baseline, and v represents the distance value vertical to the baseline, or to say the distance between the point and the baseline. In order to transform a point P from (x,y) to (s,v), it is required to finish the following steps:

- Among all points on the baseline, find out a point P', which is closest to P. Simply, the line PP' shall be vertical to the baseline.
- Find out the s value that the point P' refers to, that is the s value for point P.
- Compute the distance d between P and P', then we could choose $v = d$ or $v = -d$ according to the direction.

On the contrary, if it is required to transpose (s,v) to (x,y), we need to compute the derivative of the curve $\frac{dx}{ds}$ and $\frac{dy}{ds}$. Giving a points Q(s_0, v_0), the transformation steps are as follows:

- Find out the (x,y) values of the point Q'(s,0), note as (x_0, y_0);
- Compute $\frac{dx}{ds}$ and $\frac{dy}{ds}$ while $s = s_0$, and therefore obtain a uniform vertical vector $(\frac{-dy}{ds}, \frac{dx}{ds})$ or opposite.
- The final coordinate for point P shall be $(x_0 + v_0 \times \frac{-dy}{ds}, y_0 + v_0 \times \frac{dx}{ds})$

2.3 Path Generation

All processes of path generation are done in the coordinate (s,v). As for same reason to baseline creation, is is also preferred to create a path by cube spline curve fitting. However, in this step we don't have 4 waypoints for generating the path. We only have the current position of the vehicle O(s_1, v_1), the target point on the route T(s_2, v_2), and the direction of the vehicleθ. In general, the path should be eventually parallel to the route, which means $\frac{dv}{ds} = 0$. The curve fitting for a cubic curve $v = f_3(s)$ requires 4 variables, we could create this curve via following conditions:

$$\begin{aligned} f_3(s_0) &= v_0; \\ f_3(s_1) &= v_1; \\ \frac{dv}{ds}(s_0) &= tan(\theta); \\ \frac{dv}{ds}(s_1) &= 0. \end{aligned} \tag{2}$$

By these four conditions we could generate a path for one vehicle and one target waypoint. For the selection of path, in this step, it is required to choose multiple target waypoints and generate multiple paths.

2.4 Selection

The position of obstacles could also be computed in the Curvilinear space (s,v). Simply, by justifying whether a path goes through the obstacle, we could confirm that if a path is blocked or not. By confirming that for all paths, we could obtain a series of 1 and 0, where 1 means blocked and 0 means not. Afterwards, the authors applies Gaussian Convolution toward this series to make the result more continue. Figure. 3 shows the process. With the help of the Gaussian convolution we could avoid path through the obstacles or the path too close to the obstacles.

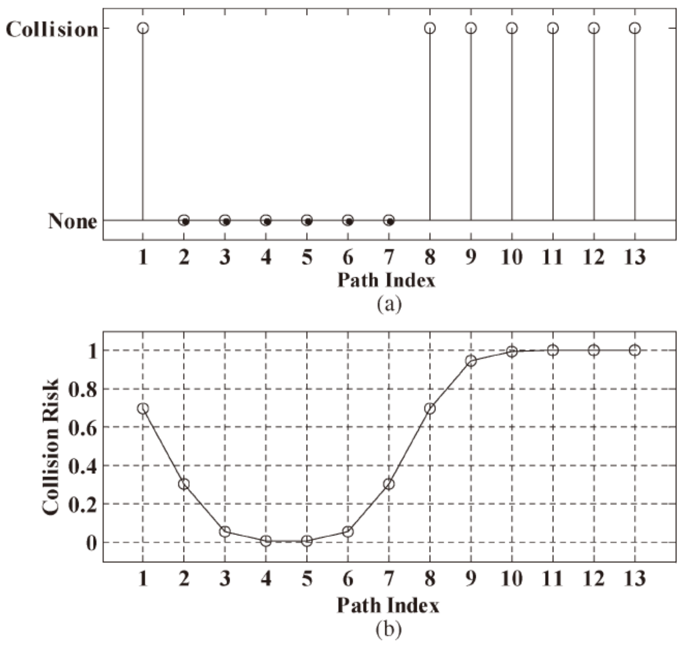

Figure 3: Path selection via Gaussian Convolution

Eventually, for all path which are not blocked, it is required to choose an optimized path. Generally, the 'Smoothest path' shall be chosen. The authors use the following formula to compute the smoothness of the route:

$$C_{k(i)} = \int k_i^2(s)ds_p \tag{3}$$

Where k_i is the curvature of a path with index i and s_p represents the length along the path. In the implementation in this paper, we used a simpler method, which is also described in the paper as 'Cost of Consistency':

$$C_{c(i)} = \frac{1}{s_2 - s_1} \int_{s_1}^{s_2} l_i ds \tag{4}$$

Where l_i is the Euclidean distance between a point in a current path candidate of index i and a point in the path of the previous step with the same arc length of the base frame. Eventually among all available path, we choose the path with minimum 'Cost of Consistency'.

2.5 Speed Control

The author also presented a speed control method which manage the speed of the vehicle according to the chosen path. As for the complexity for building a dynamic system for vehicles and limitation of time, we did not implement all process of speed control. Simply speaking, the authors compute the expected linear velocity and angular velocity according to the path chosen and the current speed. Further detailed process are described in the paper.

3 Implementation

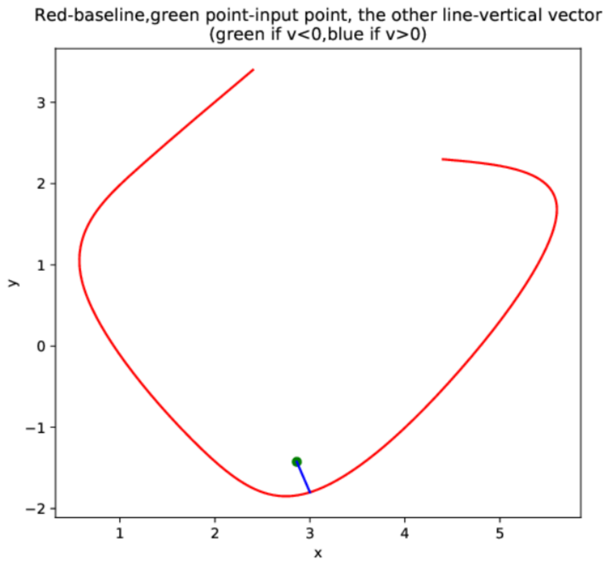

Figure 4: Result of a transform of a point(Green) from (x,y) to (s,v). The blue line shows the vertical vector.

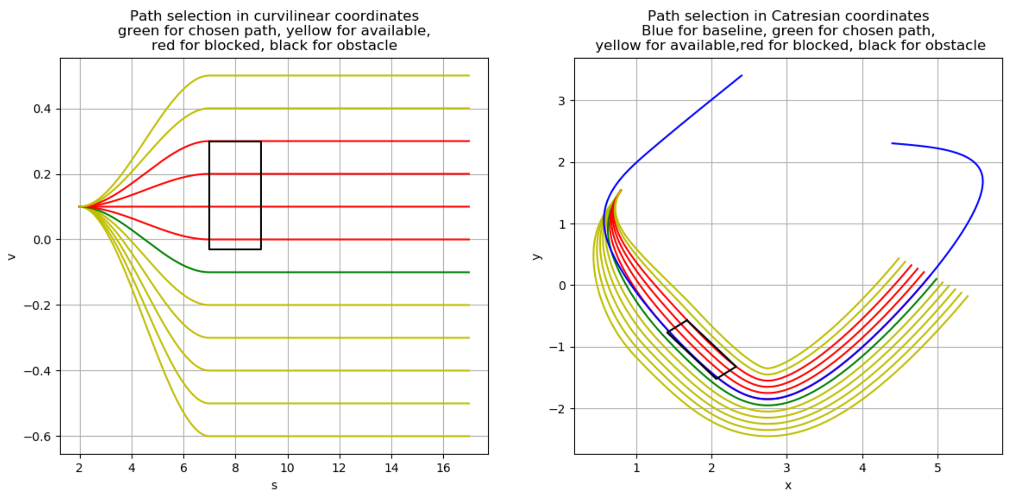

Figure 5: One simple experimental result for path choosing.

In this part, we implement the basic process of the path selection, which include baseline generation, localization, coordinate transform, path generation and path selection. The main difference between the implementation and the theory is that, a curve in computer is stored by a set of points with very small distance other than a cubic equation. This makes the implementation a little bit different to the theory, but also makes it easier to compute the derivatives as well as the vertical vectors. Initially, we choose some reasonable points to generate a route, which is the same as Figure. 2. Afterwards, we create an obstacles on the route and we input the initial position and speed of the vehicle. The whole process is generated in the codes. *readme.md* shows how to run all codes. For a simple test, we could simply run *pathchoosing.py* with python or run *Tracing.ipynb* on Jupyter notebook.

With the help of transformations, we could generate different path and choose the most suitable path according to all conditions. Figure.5 shows one result.

As for matter of time, we did not implement all process in a dynamic form. However, with the input data, we could see that the path is correctly chosen as in the paper. We've found that the path choosing in (s,v) space is quite fast, but it cost long time to turn it back onto (x,y) space. It might cause problems in real-time processing for real-time process requires to determine the velocity according to parameters in (x,y) space eventually.

4 Conclusion

We simply analyse the method used in the paper ‘Local Path Planning for Off-Road Autonomous Driving With Avoidance of Static Obstacles’, and implement it in a easy way. The process is effective especially for vehicles on route, and could possibly help the vehicles to choose their paths while driven automatically. One obvious problem is that the whole process of this method will cost some time, which might cause difficulty for real-time processing. More optimization methods needs to be applied to improve this method.