# motion planning and motion control framework for autonomous vehicles


**Abstract**

In this project, we present a motion planning and motion control framework for aggressive driving of an autonomous vehicle, with the aim of obtaining an time-optimal trajectory and control the vehicle to follow this trajectory. The framework consists of four major works: (i) a sampling-based planner, (ii) a novel path smoothing method, (iii) a trajectory optimization procedure on a track formed by smoothed path, and (iv) a trajectory tracking controller. A sampling-based planner called `Bi-iSST` is proposed to quickly generate a trajectory for autonomous driving scenario by utilizing the bidirectional approach to speed up the searching process, and avoiding solving boundary value problem. A classic smoothing approach first is applied on the path generated by the sampling-base planner, without the consideration of the dynamics and the initial orientation. Then the smoothed path is treated as a beam with supports on both ends and pre-defined stretches. Two kinds of virtual force are applied on the beam: (i) force due to the virtual potential emitted by obstacles and (ii) rebound torques if the curvature exceeds minimal turning radius. After the smooth process is finished, the beam is fitted by a parametric spline and used as the center-line of a race track in the third stage. Finally an optimization strategy is employed to get a trajectory with minimal traveling time and an MPC is applied to track this trajectory.

Another sampling-based planning algorithm called `Ref-iSST` is also developed for wireless external actuation micro-scale wire control system. A transition matrix similar to that in Markov Decision Processes(MDP) is used to form the reference trajectory.


## I. Introduction

The last three decades have seen steadily increasing research efforts, both in academia and in industry, towards developing driverless vehicle technology. Driverless cars are essentially autonomous decision-making systems that process a stream of observations from on-board sensors such as radars, LIDARs, cameras, GPS units, and odometry. These observations, together with prior knowledge about the road network, rules of the road, vehicle dynamics, and sensor models, are used to automatically select values for controlled variables governing the vehicle's motion. Intelligent vehicle research aims at automating as much of the driving task as possible. These developments have been fueled by recent advances in sensing and computing technology. The automotive engineers today face the challenge of designing autonomous safety systems that can utilize the full capabilities of the vehicle's tires in emergency scenarios to avoid accidents and significant understeer or oversteer [1] [2].

The decision making in contemporary autonomous driving systems is typically hierarchically structured into route planning, behavioral decision making, local motion planning band feedback control [3]. The partitioning of these levels are, however, rather blurred with different variations of this scheme occurring in the literature. This research focus on these core problems of autonomous aggressive driving. Particular emphasis is placed on methods for motion planning and control.

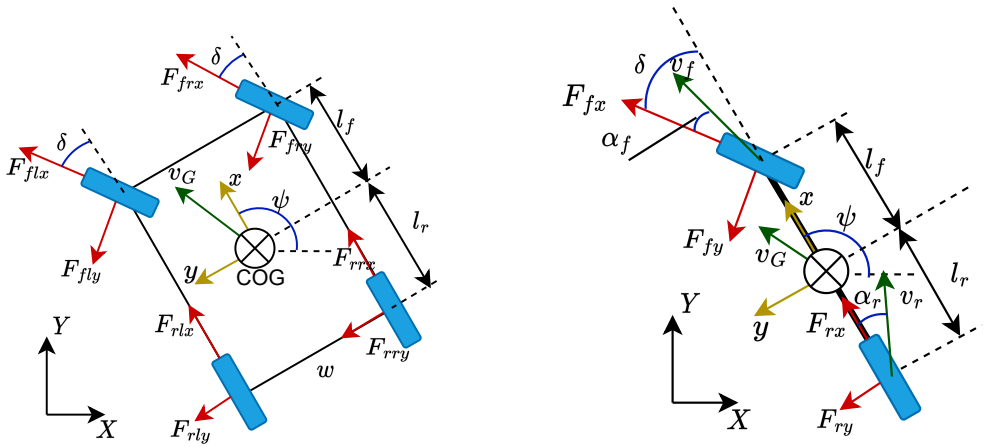


Fig. 1: Schematics of the vehicle dynamics modeling configuration.
(a) 4 wheel model; (b) 2 wheel model

## II. Problem Statement

The primary focus of this project is to develop a motion planning algorithm to find a feasible, smooth and optimal trajectory within an acceptable time for an autonomous vehicle, by utilizing the extreme performance of a vehicle, as well as a motion control algorithm that allows the vehicle to drive at the handling limits with the same capability as a professional race driver.

### A. Vehicle Model

We consider an all-wheel steering vehicle on the horizontal plane and Fig.1(a) illustrates the schematic of the vehicle modeling configuration. A ground-fixed frame $\mathcal{N}(X, Y)$ and a body-fixed frame $\mathcal{B}(x, y)$ are used to build the model. The origin of $\mathcal{B}$ is located at the vehicle mass center (denoted as $COG$). The longitudinal and lateral forces are denoted as $F_{ijx}$ and $F_{ijy}$, $i = f, r$ and $j = l, r$, at the front (rear), left (right) wheels, respectively. Denoting the steering angle as $\delta$, the equations of motion of the vehicle are obtained as:

$$\begin{aligned} m\dot{v_x} &= F_{rlx} + F_{rrx} + (F_{flx} + F_{frx})\cos(\delta) - (F_{fly} + F_{fry})\sin(\delta) + mv_y\dot{\psi} \\ m\dot{v_y} &= F_{rly} + F_{rry} + (F_{flx} + F_{frx})\sin(\delta) + (F_{fly} + F_{fry})\cos(\delta) - mv_x\dot{\psi} \\ I_z\dot{\omega} &= [(F_{fly} + F_{fry})\cos(\delta) + (F_{flx} + F_{frx})\sin(\delta))]\, l_f - (F_{rly} + F_{rry})l_r + \\ &\quad \frac{w}{2}[(F_{frx} - F_{flx})\cos(\delta) + (F_{fly} - F_{fry})\sin(\delta) + (F_{rrx} - F_{rlx})] \end{aligned} \tag{1}$$

Due to the symmetry of vehicle along the longitudinal center line, the model is simplified to a bicycle model [4]. By introducing steering speed $\delta_{\texttt{dot}}$, the Eq.(1) can be written as:

$$\dot{X} = v_x\cos(\psi) - v_y\sin(\psi) \tag{2a}$$
$$\dot{Y} = v_x\sin(\psi) + v_y\cos(\psi) \tag{2b}$$
$$\dot{\psi} = \omega \tag{2c}$$
$$\dot{v_x} = \frac{1}{m}\left(F_{rx} + F_{fx}\cos(\delta) - F_{fy}\sin(\delta) + mv_y\omega\right) \tag{2d}$$
$$\dot{v_y} = \frac{1}{m}\left(F_{ry} + F_{fx}\sin(\delta) + F_{fy}\cos(\delta) - mv_x\omega\right) \tag{2e}$$
$$\dot{\omega} = \frac{1}{I_z}\left((F_{fy}\cos(\delta) + F_{f,x}\sin(\delta))l_f - F_{ry}l_r\right) \tag{2f}$$
$$\dot{\delta} = \delta_{\texttt{dot}} \tag{2g}$$

Rewrite Eq.(2) to

$$\dot{x} = f(x, u) \tag{3}$$

where $x \in \mathcal{X}$ is the state, belonging to a n-dimensional manifold $\mathcal{X}$(the state space), and $u = [\delta_{\texttt{dot}}, F_{fx}, F_{rx}]^T$ is the control input. The preceding formulation can include both nonholonomic and dynamic constraints.

The vehicle uses rubber tires and the motion heavily depends on the tire-road interaction forces, the model of which are one of the most important part of the dynamics. The longitudinal friction forces are functions of the slip ratios, namely, $F_{ix} = \mu_x F_{i,z}, i = f, r$ where $\mu_x$ is the longitudinal friction coefficient, which is a function of the longitude slip ratios $\lambda_i$. Similarly, The lateral friction forces are functions of the slip angle, namely, $F_{ix} = \mu_y F_{iz}, i = f, r$ with lateral friction coefficient $\mu_y(\alpha_i)$, where $\alpha_i, i = f, r$ are the slip angles shown as in Fig.1(b).

The coupling effect between the longitudinal and lateral friction forces is modeled as the friction circle. With $F_{ix}$ and $F_{iy}$, the friction circle is captured as $\sqrt{F_{ix}^2 + F_{iy}^2} < \mu F_{iz}$, where $\mu$ is the total friction coefficient. For both $\mu_x$ q and $\mu_y$, we use the Pacejka's magic formula [5]

$$\mu(x) = D_t\sin\left(C_t\tan^{-1}\left(B_t(1 - E_t)x + E_t\tan^{-1}(B_t x)\right)\right) \tag{4}$$

where the variable is $\lambda_i$ or $\alpha_i$, and model parameters $B_t$, $C_t$, $D_t$, and $E_t$ are obtained through matching with experiments.

For most regular ground surface, the value of parameter $E_t$ is between 0.97 and 1.0 [6], to speed up the computation, the Eq.(4) for lateral friction coefficient then can be simplified as:

$$\mu(x) = D_t\sin\left(C_t\tan^{-1}B_t x\right) \tag{5}$$

Due to requirement of the computational speed, the longitudinal force of the wheel $F_{ix}, i = f, r$ is modeled using a motor model for the DC electric motor as well as a friction model for the rolling resistance and the drag [7]. And to exploit the

performance of the car, each wheel is installed with a brake than can be controlled independently. So the longitudinal force is:

$$F_{ix} = (C_{m1} - C_{m2}v_x)d - C_r - C_d v_x^2 - C_b b_i \tag{6}$$

where $i = f, r$ denotes the front and rear wheel, $d$ is the throttle of the motor, $C_{m1}, C_{m2}, C_r, C_d, C_b$ are performance related parameters. $b$ is the brake torque applied on wheel. According to Eq.(2) and Eq.(6), now the control input can be expressed as $u = [\delta_{\texttt{dot}}, d, b_f, b_r]^T \in \mathcal{U}$.

### *B. Motion Planning*

The motion-planning problem can be stated as follows: given an initial state $x_0 \in \mathcal{X}$ , at time $t_0$, and a goal equilibrium configuration $x_f$, find a set of control inputs $u : [t_0; t_f] \in \mathcal{U}$ that can drive the agent from $x_0$ to $x_f$ [8]. Although the usual formulation of the motion-planning problem is concerned only with finding a feasible trajectory, in this project we are also interested in finding a trajectory minimizing the traveling time.

Sampling-based motion planners, such as the rapidly exploring random tree (`RRT`) [9] and its variants, are widely applied for great efficiency at generating a trajectory. `RRT`s take sample points in the search space and add them to a tree that covers the whole space with probabilistic completeness. However, the `RRT` method is not optimal. Although the `RRT`$*$ algorithm can be asymptotically optimal [10], its convergence is slow for large scale problems. The `RRT` methods samples in state space, a process to connect current state and a new sampled state is required, which essentially is to solve a boundary value problem(BVP), leading to a huge computational consume.

### *C. Motion Control*

In order to execute the reference path or trajectory from the motion planning system, a feedback controller is used to select appropriate actuator inputs to carry out the planned motion and correct tracking errors. The tracking errors generated during the execution of a planned motion are due in part to the inaccuracies of the vehicle model. Thus, a great deal of emphasis is placed on the robustness and stability of the closed loop system. Many effective feedback controllers have been proposed for executing the reference motions to stabilize the reference path or trajectory in the presence of modeling error and other forms of uncertainty. Depending on the reference provided by the motion planner, the control objective may be path stabilization or trajectory stabilization.

Model predictive control (MPC) is a general control design methodology which can be very effective for autonomous driving. Conceptually, the approach is to solve the motion planning problem over a short time horizon, take a short interval of the resulting open loop control, and apply it to the system. While executing, the motion planning problem is re-solved to find an appropriate control for the next time interval. Advances in computing hardware as well as mathematical programming algorithms have made predictive control feasible for real-time use in driverless vehicles.

## III. Completed Work

Fig.2 shows an overview of the motion planning and tracking scheme of the project. Different from other frameworks which use the path directly from sampling-based algorithm (and a smooth process) as the reference trajectory, in this framework the smoothed path is used as the centerline of a racetrack, and an optimization process called Minimum-Lap-Time is applied to get a time-stamped optimal trajectory on this road. It is a big challenge to get the optimal control input sequence for a racing car due to its high speed and nonlinear behavior of the high fidelity model.

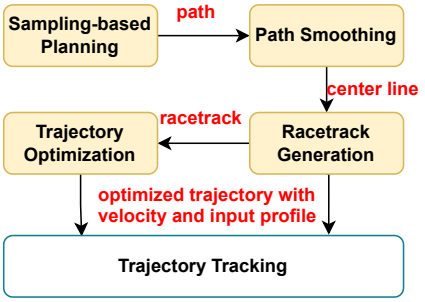


Fig. 2: Path Planning Framework

### A. Bi-Directional Sparse Stable Sampling-Based Algorithm

The RRT samples in state space, then tries to connect the sampled state to the nearest node on the tree by solving a boundary value problem. The Stable Sparse RRT(SST) [11] samples on input space to avoid such extra computational cost. However, sampling in input space breaks the uniform distribution in state space, leading to a lot of redundant samplings. To solve this problem, an additional set, $\mathbb{W}$, is employed in SST to monitor the sampling states, to guarantee in a certain region of the state space only one node, which has least cost from the start state (in most cases it refers to the traveling time), is active at most. Any new state located into this region has to compare the cost with this active node. The states with higher cost will be ignored.

The bidirectional informed SST, Bi-iSST, is developed for this project. Algorithm 1 shows the structure of the Bi-iSST algorithm. Compared with the SST algorithm, the Bi-iSST algorithm maintains two trees, $\mathcal{G}_\mathtt{a}$ and $\mathcal{G}_\mathtt{b}$, as well as their witness sets, $\mathbb{W}_\mathtt{a}$ and $\mathbb{W}_\mathtt{b}$. The roots of the two trees are on the start state and the target state of all the agents, respectively. A tree is randomly selected to propagate by following the iSST propagation as described in Algorithm 2, which generates a set of new vertices $\mathbb{X}$.

The 'i' in 'iSST' means informative. By using the information of the target, we can control the tree propagation direction, to speed up the path finding process.

A set $\mathbb{P}$ is maintained to collect the vertices that are near to any vertex whose distance to the other tree is below $\delta_\mathtt{watch}$. A process of solving BVP is employed to connect the vertices in $\mathbb{P}$ to the other tree, to accelerate the path finding speed.

**Algorithm 1:** Bi-iSST

$\mathcal{G}_\mathtt{a}, \mathbb{W}_\mathtt{a} \leftarrow \mathtt{init}(\mathbf{q}_\mathtt{s})$, $\mathcal{G}_\mathtt{b}, \mathbb{W}_\mathtt{b} \leftarrow \mathtt{init}(\mathbf{q}_\mathtt{t})$, $\mathbb{P} \leftarrow \emptyset$
**for** $m$ *iterations* **do**
  $u \leftarrow \mathtt{Uniform}([0, 1])$
  **if** $u < 0.5$ **then**
    $\mathbb{X} \leftarrow \mathtt{iSST\_Propagation}(\mathcal{G}_\mathtt{a}, \mathbb{W}_\mathtt{a}, \mathbf{q}_\mathtt{t})$
    $\mathcal{G}' \leftarrow \mathcal{G}_\mathtt{b}$
  **else**
    $\mathbb{X} \leftarrow \mathtt{iSST\_Propagation}(\mathcal{G}_\mathtt{b}, \mathbb{W}_\mathtt{b}, \mathbf{q}_\mathtt{s})$
    $\mathcal{G}' \leftarrow \mathcal{G}_\mathtt{a}$
  **foreach** $x \in \mathbb{X}$ **do**
    $x_\mathtt{closest} \leftarrow \mathtt{Closest\_Node}(\mathcal{G}', x)$
    **if** $\mathtt{distance}(x, x_\mathtt{closest}) < \delta_\mathtt{watch}$ **then**
      $\mathbb{P} \leftarrow \mathbb{P} \cup \{x\}$

**Algorithm 2:** iSST_Propagation

$x_\mathtt{curr} \leftarrow \mathtt{Search\_Selection}(\mathbb{O}, \mathbb{O}')$, $\mathbb{X} \leftarrow \emptyset$
**while** *True* **do**
  $x_\mathtt{new} \leftarrow \mathtt{Blossom}(x_\mathtt{curr}, q)$
  **if not** $\mathtt{Collision\_Free}(\overline{x_\mathtt{curr} \rightarrow x_\mathtt{new}})$ **then**
    **return**
  **if not** $\mathtt{Local\_Best\_SST}(x_\mathtt{new}, \mathbb{W}, \delta_\mathtt{s})$ **then**
    **return**
  $\mathbb{X} \leftarrow \mathbb{X} \cup x_\mathtt{new}$, $\mathbb{V}_\mathtt{active} \leftarrow \mathbb{V}_\mathtt{active} \cup \{x_\mathtt{new}\}$
  $\mathbb{E} \leftarrow \mathbb{E} \cup \{\overline{x_\mathtt{curr} \rightarrow x_\mathtt{new}}\}$
  $\mathtt{Prune\_Dominated\_Nodes\_SST}(\mathcal{G}, \mathbb{W}, x_\mathtt{new})$
  $x_\mathtt{curr} \leftarrow x_\mathtt{new}$

### B. Trajectory Smoothing

Several sampling-based motion planners are able to output an optimal trajectory [11] [12]. However, massive computation time is required to achieve such optimal trajectory. For real applications, the time for planners is limited, causing jerky output paths which are less human-friendly. Typically, the output of the sampling-based planner should be smoothed. However, in

most smooth methods the dynamics feasibility will not be maintained and are not applicable for car-like robot since the initial orientation is not considered. For example the Convex Elastic Smoothing (CES) [13] algorithm (CES) algorithm uses the following strategy:

$$\min \quad \sum \|D_{i+1} - D_i\|_2^2 \tag{7a}$$

$$\text{s.t.} \quad D_k = \pi_i - \pi_{i-1}, \pi_1 = Q_1, \pi_n = Q_n \tag{7b}$$

$$\pi_i \in \mathcal{P}_i, \|D_{i+1} - D_i\| \leqslant \frac{d^2}{R_{\min}} \tag{7c}$$

$$\pi_2 = P_1 + k\frac{Q_2 - Q_1}{\|Q_2 - Q_1\|} \tag{7d}$$

Where $\mathcal{P}$ is a collision-free tube along the original trajectory $Q$, and $\pi$ is the smoothed path. Eq.(7d) is used to control the initial orientation of the smoothed path. However, in this case, the sharp turn may appear at the second segment $D_2$ sometimes.

The proposed approach here aims at solving such phenomenon and to generate a more natural path. Take the trajectory from sampling-based method as a Bernoulli-Euler [14] beam with some external forces and pre-defined deformations, as well as constraints on the two ends. The dynamics deforming process due to such forces and deformations is equivalently treated as the path smoothing process. Now take the dynamic analysis of the beam by deriving its equations of motion(EOMs) to get the variation of displacements with time, by taking into account of the effects of inertia and viscous damping.

The internal potential energy of the beam can be described as:

$$U = \frac{1}{2}\int_V \sigma\varepsilon dV \tag{8}$$

and the total kinetic energy is:

$$T = \frac{1}{2}\int_V \rho v^2 dV \tag{9}$$

The finite element method by dividing the beam to $N$ elements and decoupling the stretching and bending process is adopted to solve the problem. The total internal potential energy for each element can be written as:

$$U^e = \frac{1}{2}EA\int_0^{L^e}\left(\frac{du}{dx}\right)^2 dx + \frac{1}{2}EI\int_0^{L^e}\left(\frac{d^2v}{dx^2}\right)^2 dx \tag{10}$$

where $u$ and $v$ are axial and transverse displacement of the beam. $E$,$A$ and $I$ are young's modulus, section area and moment of inertia, respectively. By denoting $\rho$ as the density, the kinetic energy can be expressed as:

$$\begin{aligned} T^e &= {T_s}^e + {T_b}^e \\ &= \frac{1}{2}\int_0^{L^e}\rho A\left(\frac{du}{dt}\right)^2 dx + \frac{1}{2}\int_0^{L^e}\rho A\left(\frac{dv}{dt}\right)^2 dx \end{aligned} \tag{11}$$

by ignoring the rotation energy. The first items of $U^e$ and $T^e$ are the energies due to stretching while the second items are due to bending.

*1) Stiffness and Mass Matrix:* Consider the type of the finite element as a 2-node beam-column, each node of which has 3 degrees of freedom(DOFs) and the displacement vector is:

$$\boldsymbol{w} = [u_1, v_1, \theta_1, u_2, v_2, \theta_2]^T \tag{12}$$

where $\theta$ is the bending angle. The subscript 1 and 2 represents the two ends of the element respectively.

Apply the shape functions

$$Z_{u1} = 1 - \zeta_s, Z_{u2} = \zeta_s \tag{13}$$

for stretching, and Hermite polynomials shape functions

$$\begin{aligned} Z_{v1} &= 1 - 3{\zeta_b}^2 + 2{\zeta_b}^3 \\ Z_{v2} &= 3{\zeta_b}^2 - 2{\zeta_b}^3 \\ Z_{\theta 1} &= L(\zeta_b - 2{\zeta_b}^2 + {\zeta_b}^3) \\ Z_{\theta 2} &= L(-{\zeta_b}^2 + {\zeta_b}^3) \end{aligned} \tag{14}$$

for bending where $\zeta_s = (x - u_1)/L$ and $\zeta_b = (2x - L)/L$, the symmetric stiffness matrix $\mathbf{K_e}$ and mass matrix $\mathbf{M_e}$ can be derived by putting the functions $\chi = \sum_{i=1}^{2} Z_{\chi i}$,$(\chi = u, v, \theta)$ to potential energy expression Eq.(10) and Eq.(11) respectively:

$$\mathbf{K_e} = \frac{E}{l}\begin{bmatrix} A & 0 & 0 & -A & 0 & 0 \\ & 12I/l^2 & 6I/l & 0 & -12I/l^2 & 6I/l \\ & & 4I & 0 & -6I/l & 2I \\ & & & A & 0 & 0 \\ & & & & 12I/l^2 & -6I/l \\ & & & & & 4I \end{bmatrix} \tag{15a}$$

$$\mathbf{M_e} = \frac{\rho A l}{420}\begin{bmatrix} 140 & 0 & 0 & 70 & 0 & 0 \\ & 156 & 22l & 0 & 54 & -13l \\ & & 4l^2 & 0 & 13l & -3l^2 \\ & & & 140 & 0 & 0 \\ & & & & 156 & -22l \\ & & & & & 4l^2 \end{bmatrix} \tag{15b}$$

*2) External Force:* The vehicle is a non-holonomic system and the initial orientation $\bar{\theta}_0$ shall be considered during the smoothing process, which is corresponding a fixed support on the end of the beam. Here the final orientation of the vehicle is not considered, which leads to a pinned support in the beam simulation.

The deformation of the beam is restricted by the environment obstacles. Assume there are virtual potential fields emitted by obstacles and the force applied on the beam due to such fields is expressed as:

$$\boldsymbol{F_o} = \sum_{i=1}^{N_o} \left( \frac{k_b}{\|\boldsymbol{l_i}\|^2} \cdot \frac{\boldsymbol{l_i}}{\|\boldsymbol{l_i}\|} \right) \tag{16}$$

Where $k_b$ is the intensity of the potential field and $\boldsymbol{l_i}$ is the displacement vector from obstacle to beam node. The aforementioned force pushes the beam away from obstacles, causing the beam unable to traverse any obstacle, leading to the smoothing process restricted in local tunnel.

The final shape of the beam is used as a center-line of a racetrack. The following condition should be satisfied at anywhere of the center-line:

$$n\kappa < 1 \tag{17}$$

where $n$ is the maximum allowed normal offset while $\kappa$ is the curvature. To allow the vehicle passing through, the minimum normal offset should be controlled, i.e., the maximum allowed curvature $\kappa_{\texttt{max}}$ is limited. Hence, a virtual torque proportional to the curvature is applied to the beam element:

$$T_m = \begin{cases} 0, & \text{if } \kappa < \kappa_{max} \\ -k_m \kappa, & \text{if } \kappa > \kappa_{max} \end{cases} \tag{18}$$

$k_m$ is a constant coefficient, and the negative sign implies the torque is opposite to the bending direction.

*3) Modeling:* After modeling and meshing the beam, we may get the global mass matrix $\mathbf{M}$ and stiffness matrix $\mathbf{K}$ by assembling the matrices of the elements in Eq.(15) together using the frame transform matrix as described in [15], and the assembled DOF is $\boldsymbol{d} = [d_0^x, d_0^y, d_0^\theta, d_1^x, d_1^y, d_1^\theta, ..., d_N^x, d_N^y, d_N^\theta]^T$.

Among the five constraint DOFs $d_0^x, d_0^y, d_0^\theta, d_N^x$ and $d_N^y$, the translation displacements of the first node are zeros while the others are not. The force due to pre-defined deformation on those nodes should be considered:

$$\boldsymbol{F}_d = \mathbf{K}\boldsymbol{d}_p \tag{19}$$

where $\boldsymbol{d}_p = [0, 0, \bar{\theta}_0, 0, 0, 0, ..., \bar{x}_N, \bar{y}_N, 0]$ is the constrained displacement.

Without damping, the system tends to oscillate. A proper viscous damping can avoid the oscillation and decrease the settling time. Since mass matrix $\mathbf{M}$ and stiffness matrix $\mathbf{K}$ are positive definite, the critical damping matrix $\mathbf{C}$ can be calculated by the Inman and Andry method [16]:

$$\mathbf{C} = 2\left(\mathbf{M}^{-1/2}\mathbf{K}\mathbf{M}^{-1/2}\right)^{1/2} \tag{20}$$

The final EOM can be expressed as:

$$\mathbf{M}\ddot{\boldsymbol{d}} + \mathbf{C}\dot{\boldsymbol{d}} + \mathbf{K}\boldsymbol{d} = \boldsymbol{F}_o + \boldsymbol{T}_m - \boldsymbol{F}_d \tag{21}$$

*4) simulation:* An integrator is used to simulate the stress release process by integrating on Eq.(21), with initial state $\boldsymbol{d}_0 = [0, 0, \bar{\theta}_0, \bar{x}_1, \bar{y}_1, \bar{\theta}_1, ..., \bar{x}_N, \bar{y}_N, \bar{\theta}_N]^T$ and constraints $d_0^x = 0$, $d_0^y = 0$, $d_0^\theta = \bar{\theta}_0$, as well as $d_N^x = \bar{x}_N$, and $d_N^y = \bar{y}_N$.

Note the proposed FEM method can't be applied to a real beam since deformation of the virtual beam is large, and Eq.(10),(11),(13) and (14) are all based on the assumption of small deformation. Readers can refer [17] for theories of large deformation beam, or [18] [19] for elastica theory.

### C. Global Trajectory Optimization

*1) Track Generation:* The curve of the refined beam from the previous section can be used as a smoothed path directly. However, to get an more optimal trajectory with velocity and control input profiles, we form a racetrack with width and piece-wise polynomials is used to fit the curve. Suppose the cubic curve $\mathbf{\Gamma}_i$ fit the shape of $i$th beam element [20]

$$\mathbf{\Gamma}_i(\tau) = \sum_{j=0}^{3} \binom{j}{3} \tau^j (1-\tau)^{3-j} \cdot P_{ij} \tag{22}$$

for $i = 0, ...N-1$, where $P_{i0} = e_i^1$ and $P_{i3} = e_i^2$ are the coordinates of the two nodes of the element $e_i$. Note $\mathbf{\Gamma}_i(1) = \mathbf{\Gamma}_{i+1}(0) = P_{(i+1)0} = P_{i3}$. $P_{i1}$ and $P_{i2}$ are two control points we are going to find according to the following $\mathcal{C}^2$ smooth conditions [21]:

$$\begin{aligned} \mathbf{\Gamma}'_{i-1}(1) &= \mathbf{\Gamma}'_i(0), \\ \mathbf{\Gamma}''_{i-1}(1) &= \mathbf{\Gamma}''_i(0), i = 2, ..., n \end{aligned} \tag{23}$$

Eq.23 provides $4(N-1)$ equations. The final pose of the car is not taken into consideration, so another boundary condition which provides 2 equations can be introduced:

$$\mathbf{\Gamma}''_{N-1}(1) = 0 \tag{24}$$

If we also apply natural boundary condition on the initial point:

$$\mathbf{\Gamma}''_0(0) = 0 \tag{25}$$

then a typical natural cubic spline can be fit by solving those $4n$ linear equations. However, we'd prefer the the track to align with the orientation of the car at initial state, to tap the accelerating potential, which means the direction of the curve at $\mathbf{\Gamma}_0(0)$ should be the same as the initial orientation, so we can use the following boundary condition to replace Eq.(25):

$$\frac{dy}{dx}|_{\mathbf{\Gamma}_0(0)} = \tan(\psi_0) \tag{26}$$

where $\psi_0$ is the initial orientation of the car. However, Eq.(26) only provides one equation, we still need to seek another equation to get all $P_{i1}$ and $P_{i2}$. Or we may optimize a curve similar to natural cubic spline, but the constraint is released from Eq.(25) to Eq.(26), which can be summarized as a QCQP problem:

$$\begin{aligned} \min \quad & \|\mathbf{\Gamma}''_0(0)\|_2^2 + \beta \sum_{i=0}^{n-1} (\|P_{i1} - \bar{P}_{i1}\|_2^2 + \|P_{i2} - \bar{P}_{i2}\|_2^2) \\ \text{s.t.} \quad & \text{Eq.(23), Eq.(24), Eq.(26)} \end{aligned} \tag{27}$$

where $\beta$ is the weight, $\bar{P}_{i1}$ and $\bar{P}_{i2}$ are coefficients for natural cubic spline curve with same waypoints. The first term of the objective function can be considered as the moment due to the fixed support. And the second term represents the similarity to a natural cubic spline.

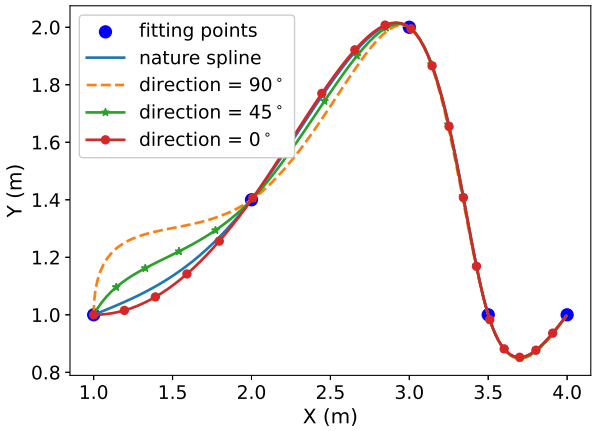


Fig. 3: Directional Spline

Fig.3 shows splines with same fitting points but different initial orientations.

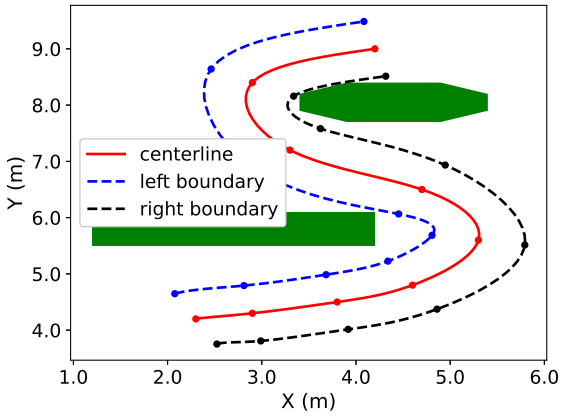

Fig. 4: A Sample of Racetrack

Now we may use to the following parametric curve equation to represent the center-line of the track by simplifying Eq.(22):

$$\mathbf{\Gamma}(\tau) = \sum_{j=0}^{3} \binom{j}{3} (\tau - \lfloor \tau \rfloor)^j (1 - (\tau - \lfloor \tau \rfloor))^{3-j} \cdot P_{\lfloor \tau \rfloor j} \tag{28}$$

where $\tau \in [0, N]$ is the parameter of the curve.

The same approach can be applied to generate the left boundary parametric curve $\mathbf{\Upsilon}(\tau)$ and right boundary $\mathbf{\Psi}(\tau)$ for the racetrack. The waypoints of the left boundary can be derived from the center-line $\mathbf{\Gamma}(\tau)$:

$$W_{li} = \min\left(n_{\texttt{max}}, n_i\right) \cdot \mathbf{T}_l \frac{\mathbf{\Gamma}'(i)}{\|\mathbf{\Gamma}'(i)\|_2} \tag{29}$$

where $n_{\texttt{max}}$ is the maximum allowed track offset while $n_i$ is the minimum distance from $\mathbf{\Gamma}(i)$ to obstacles along the normal direction, and $\mathbf{T}_l = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ is the rotation transform matrix. Same treatment can be applied to get the right boundary $W_{ri}$ by replacing $\mathbf{T}_l$ with $\mathbf{T}_r = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$. Such treatment approximately makes the vectors $\overrightarrow{\mathbf{\Gamma}(\tau)\mathbf{\Psi}(\tau)}$ and $\overrightarrow{\mathbf{\Gamma}(\tau)\mathbf{\Upsilon}(\tau)}$ perpendicular to $\mathbf{\Gamma}'(\tau)$ for any $\tau \in [0, \tau_{\texttt{max}}]$.

Fig.4 show an example of a racetrack.

*2) Trajectory Optimization:* The purpose of the trajectory optimization is to find the vehicle control input and a feasible path that minimize the total traveling time $T$ of moving the vehicle along the track from start state to the target. As explained in [22], the most effective way to describe the optimization strategy is in the Frenet-Serret(F-S) planar frame, which is shown in Fig.5, where $s$ is the longitudinal displacement along the racetrack, e.g. the traveled length of the racetrack, $n$ is the deviation from the center-line, and $\psi_c$ is the tangent angle of the curve of the center-line. The calculus of the vehicle speed in the dynamics model of Eq.(2) in F-S planar frame now is:

$$\dot{s} = \frac{v_x \cos(\psi - \psi_c) - v_y \sin(\psi - \psi_c)}{1 - n\kappa} \tag{30a}$$

$$\dot{n} = v_x \sin(\psi - \psi_c) + v_y \cos(\psi - \psi_c) \tag{30b}$$

Since the analytical expression of the center-line $\mathbf{\Gamma}$ with respect to $\tau$ is already presented in Eq.(28), $s$ in Eq.(30a) can be substituted by $\tau$ with the relations $\frac{ds}{d\tau} = |\mathbf{\Gamma}'(\tau)|$:

$$\dot{\tau} = \frac{1}{|\mathbf{\Gamma}'|} \frac{v_x \cos(\psi - \psi_c) - v_y \sin(\psi - \psi_c)}{1 - n\kappa} \tag{31}$$

Now the state vector is $\boldsymbol{x} = [\tau, n, \psi, v_x, v_y, \omega, \delta]^T$. Let $f(\boldsymbol{x}, \boldsymbol{u})$ be the dynamics of the vehicle, the optimal trajectory can be obtained by solving the following optimal control problem (OCP):

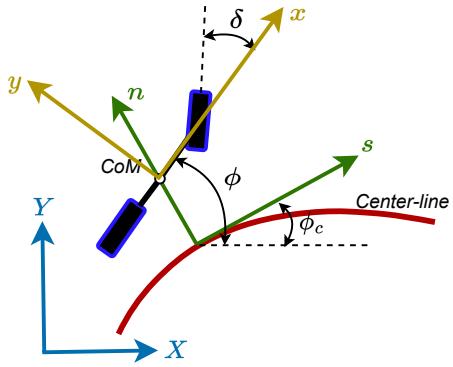


Fig. 5: Frenet-Serret Frame

$$\begin{aligned}
\min \quad & T && \text{(32a)}\\
\text{s.t.} \quad & \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) && \text{(32b)}\\
& \boldsymbol{\psi}(\delta, \delta_{dot}, d, b_f, b_r) \leqslant 0 && \text{(32c)}\\
& n^{\min}(\tau(t)) \leqslant n \leqslant n^{\max}(\tau(t)) && \text{(32d)}\\
& \boldsymbol{x}(0) = \bar{\boldsymbol{x}}_0, \tau(T) = N, v_x(T) = 0 && \text{(32e)}
\end{aligned}$$

where Eq.(32c) are algebraic inequalities that bound state variable $\delta$ and control variables $\delta_{dot}$, $d$, $b_f$ and $b_r$. Inequality (32d) is used to keep the vehicle traveling in the range of the racetrack, where $n^{\min}(\tau(t)) = -\|\boldsymbol{\Gamma}(\tau(t)) - \boldsymbol{\Psi}(\tau(t))\|$ and $n^{\max}(\tau(t)) = \|\boldsymbol{\Gamma}(\tau(t)) - \boldsymbol{\Upsilon}(\tau(t))\|$. The OCP (32e) is hard to solve due to the high non-linearity of $\boldsymbol{\Gamma}$ and the presence of inequality constraint(32c). The following two treatments are applied to increase the chance of getting an optimal solution.

1) Discrete the whole racetrack to $N^c$ segments with equivalent length.
2) Convert the inequality constraint (32d) to soft constraints and take them as a penalty term in objective function, the value of which is small when $n^{\min} \leqslant n \leqslant n^{\max}$ but increases rapidly if $n$ exceeds the boundaries.

We use the exponential as the penalty term and let $s(\overline{\tau}_{i+1}) - s(\overline{\tau}_i) = s(\overline{\tau}_i) - s(\overline{\tau}_{i-1})$ for $i \in [1, N^c]$, the OCP(32e) can be rewritten as:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{N^c} \left( \Delta t_i + \gamma \left( e^{\lambda(n_i^{\min} - n_i)} + e^{\lambda(n_i - n_i^{\max})} \right) \right) && \text{(33a)}\\
\text{s.t.} \quad & \boldsymbol{x}_{i+1} = \Delta t_i f(\boldsymbol{x}_i, \boldsymbol{u}_i) && \text{(33b)}\\
& \boldsymbol{\psi}(\delta_i, \delta_i^{dot}, d_i, b_{fi}, b_{ri}) \leqslant 0 && \text{(33c)}\\
& \tau_i = \bar{\tau}_i && \text{(33d)}\\
& \boldsymbol{x}_0 = \bar{\boldsymbol{x}}_0, v^x_{N^c+1} = 0 && \text{(33e)}
\end{aligned}$$

where $\gamma$, $\lambda$ are weights, $\Delta t_i$ is the traveling time of $i$th segment. Since $\overline{\tau}_i$ is known, other parameters, e.g. $n_i^{\min}, \psi_i^c$, related to $\tau_i$ can be gained, which simplifies the calculation of the OCP.

### *D. Trajectory Tracking(Motion Control)*

An Nonlinear MPC (NMPC) shown in Eq.(34) is applied for trajectory tracking. Different from trajectory generating process, Cartesian frame rather than F-S frame is used to simplify the computational process. The reference trajectory can be represented by $\boldsymbol{y} = tr(t), t \in [0, T]$, where $T$ is the optimal time of the trajectory and $\boldsymbol{y}$ is the vehicle state in F-S frame. We also can map the traveling distance to traveling time as $t = tr^{-1}(\boldsymbol{y})$. For a vehicle state $\widetilde{\boldsymbol{x}_0}$ in Cartesian frame, whose corresponding state in F-S frame is $\widetilde{\boldsymbol{y}_0}$, we first get the reference time $t_0 = tr^{-1}(\widetilde{\boldsymbol{y}_0})$, and divide the time span $[t_0, t_0 + T]$ to $k+1$ reference time points. Then get a set of reference states $\widetilde{\boldsymbol{y}_0}, \widetilde{\boldsymbol{y}_1}, ..., \widetilde{\boldsymbol{y}_k}$ and $\widetilde{\boldsymbol{x}_0}, \widetilde{\boldsymbol{x}_1}, ..., \widetilde{\boldsymbol{x}_k}$ via $\boldsymbol{y} = tr(t)$

$$\begin{aligned}
\min \quad & \sum_{i=0}^{N} \left( (\boldsymbol{x}_i - \widetilde{\boldsymbol{x}_i})^T Q (\boldsymbol{x}_i - \widetilde{\boldsymbol{x}_i}) + (\boldsymbol{u}_i - \widetilde{\boldsymbol{u}_i})^T R (\boldsymbol{u}_i - \widetilde{\boldsymbol{u}_i}) \right)\\
\text{s.t.} \quad & \boldsymbol{x}_{i+1} = \Delta t_i f(\boldsymbol{x}_i, \boldsymbol{u}_i) && \text{(34)}\\
& \boldsymbol{\psi}(\delta_i, \delta_i^{dot}, d_i, b_i) \leqslant 0\\
& \boldsymbol{x}_0 = \widetilde{\boldsymbol{x}_0}
\end{aligned}$$

where $Q$ and $R$ are weight matrices for state and control input, respectively.

### *E. The* `Ref-iSST` *algorithm and Global Route*

(Note the symbols used in this subsection are independent of those in previous sections.)

The `Ref-iSST` path planning algorithm is developed for another project - the microfluidic system that consists of $N \times N$ electrode arrays. The electrodes are independently and intermittently powered with a DC voltage. The electrode arrays are on the substrate and covered by a fluid that contains a dilute concentration of micro- or nanowires. The motion of the wires is guided by the electrical field wirelessly. One of the biggest limitations of wireless external actuation is its global and coupled influence in the workspace, which limits the capability to robustly manipulate multiple agents independently and simultaneously.

A reference trajectory is required to control the wires from their random positions to the destinations. By applying the Bi-iSST algorithm, we find the wires are prone to travel along the edge of the cell formed by the nearest $2 \times 2$ electrodes. The Ref-iSST algorithm is developed based on this phenomenon. First an optimal global reference trajectory is generated by scheduling the electrode array states. Then an algorithm is applied to force the wires following the global reference trajectory.

*1) Global Route:* The global routing here means scheduling a sequence of turning on and off of the electrodes to form the reference trajectory. Because the electric field acting on an agent is inversely proportional to the square of the distance to the electrodes, only the neighboring electrodes have significant effects on the motion of the agents. The algorithm is based on the following two assumptions:

1) An agent can only be located on an electrode and count one-step if there is any agent that moves from one electrode to a neighboring electrode.
2) In one step, an agent can go to any of the neighboring electrode by turning on the electrode where the agent locates and turning off the destination one. Or the agent can be UNMOVED by turning on the electrode where the agent locates.

For an $N \times N$ electrodes microfluidic device with $m$ micro agents, there are $N^{2m}$ states in total. A value $s = \sum_1^m (r_i N^i + c_i N^{i-1})$ is used to indicate a state, where $r_i$ and $c_i$ are the row and the column indices of the electrode where the $i$th agent locates.

Similar to the MDPs [23], a pre-constructed transition matrix $\mathbf{M}_t \in \mathcal{R}^{N^{2m} \times N^{2m}}$ is used to indicate the transition between two states. Each element in the MDP transition matrix represents the state transition probability. It can be simplified to zero or one in global routing, e.g. $\mathbf{M}_{ij} \in \{0, 1\}$, where $\mathbf{M}_{ij} = 1$ represents the $i$th state can be reached from $j$th state within one step. By monitoring the value of $j$th row of the vector $S_n$ in a continued multiplication $S_n = \mathbf{M} S_{n-1}$, we can obtain the minimum steps from a start configuration to a target when $S_n(s_t) \neq 0$. $j$ corresponds the target state and $S_0$ is the initial vector whose $s_0$th element representing the start state is set to one while others are zero, e.g. $S_0(i) = 1$ for $i = s_0$ and $S_0(i) = 0$ for $i \neq s_0$.

The GBTM (Global Routing With Transition Matrix) algorithm uses a tree to schedule the global trajectory as shown in Algorithm 3. First the initial state is set as the root of the tree and set $\mathbb{P}$ can be considered as the collection of tree leaves. In Line 2, we get the minimum steps from the start to the target states, i.e., the depth of the tree as described above. The feasible states of the next step for each leaf(Lines 5-6) can be obtained via the transition matrix. To reduce the redundancy of the tree, the states whose minimum estimated steps to target exceeds $n_{\texttt{min}} - k$ are removed, where $k$ is the depth of the current leaf. Because a new state may be conducted from different leaves, only the leaf with the best quality will be selected as the parent of the new state. The quality of a node is defined as $\sum_1^k \sum d_m(e_i, e_j)$, where $d_m$ is the Manhattan distance of the two turned "on" electrodes $e_i$ and $e_j$, $i, j = 1, \cdots, N^2$ and $i \neq j$ for each step. In Line 9-11, the leaves at $(k-1)$th step with no child are trimmed. Set $\mathbb{Y}$ contains only one leaf that is the same as $\mathbf{q_t}$ at the last step, which guarantees the existence of the path from the start to the target state.

It is infeasible to store the whole transition matrix even though the matrix is in sparse format because its size grows exponentially with the number of the agents. Such kind of matrices can be constructed by two low-dimension control transition matrices $\mathbf{D}$, whose structure is similar to the transition matrices $\mathbf{M}$ while a control matrix $\mathbf{C} \leftarrow \mathbb{R}^{N \times N}$ is used to represent the none-zero element. Each element of the matrix $\mathbf{C}$ reflects the state of an electrode. For instance, 1, -1, and 0 represent "on","off", and "free" states, respectively, where "free" refers to the state of the electrode that does not affect the transition process. The product of the control matrices is defined as

$$\mathbf{C}_1 \cdot \mathbf{C}_2 = \begin{cases} 0, \texttt{if } \mathbf{C}_1 \texttt{ and } \mathbf{C}_2 \texttt{ not compatible}, \\ 1, \texttt{if } \mathbf{C}_1 \texttt{ and } \mathbf{C}_2 \texttt{ compatible}. \end{cases} \tag{35}$$

For the previous instance, $\mathbf{C}_1 \cdot \mathbf{C}_2 = 0$ (not compatible) if any of $\mathbf{C}_1(i,j) \cdot \mathbf{C}_2(i,j) = -1$, otherwise $\mathbf{C}_1 \cdot \mathbf{C}_2 = 1$.

For a state $s$ with $m$ agents, divide the state into two sub-configurations with $m = m_1 + m_2$ and $s = s_1 N^{2m_2} + s_2$. Then apply the matrix multiplication $\widetilde{\mathbf{D}^s} = \mathbf{D}_1^{s_1} \times (\mathbf{D}_2^{s_2})^T$, where $\mathbf{D}_i^{s_i} (i = 1, 2)$ are the control transition vectors for the $m_i$ agent's

**Algorithm 3:** `GRTM`

```
1  T ← init(q_s), P ← {q_s}
2  n_min ← Find_Minimum_Steps(q_s, q_t)
3  for k ← 1 to n_min do
4      Y ← ∅
5      foreach q ∈ P do
6          Y ← Y ∪ Get_Next_States(q, n_min − k)
7      foreach y ∈ Y do
8          Set_Best_Parent(y, T, P)
9      foreach q ∈ P do
10         if Has_No_Child(q) then
11             Remove_From_Tree(T, q)
12     P ← Y
13 R ← Get_Global_Route(T)
```

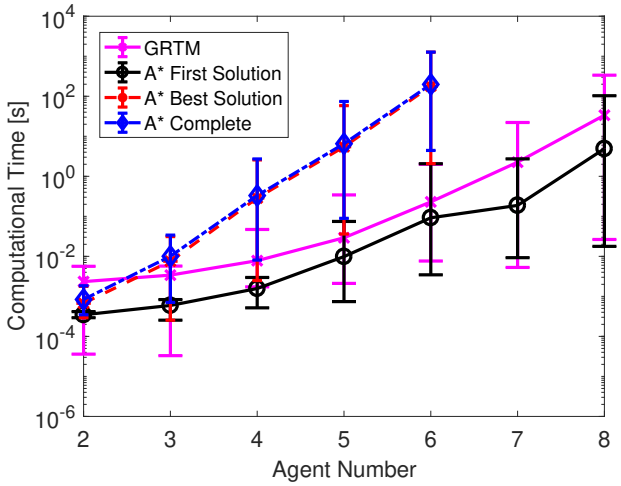


Fig. 6: Computational cost for the global routing algorithms.

configuration at state $s_i$ ($s_i$th column of matrix $\mathbf{D}_i$), and $\widetilde{\mathbf{D}^s}(i,j) = \mathbf{D}_1^{s_1}(i) \cdot \mathbf{D}_2^{s_2}(j)$. Finally, resize $\widetilde{\mathbf{D}^s} \in \mathcal{R}^{N^{2m_1} \times N^{2m_2}}$ to $\mathbf{D}^s \in \mathcal{R}^{N^{2(m_1+m_2)} \times 1}$ to get the transition vector for $m$ micro agents at the state $s$.

The comparison of the global routing algorithm and A* is shown in Fig. 6. The A* First Solution is the computational time of the very first path found by the A*-based global routing algorithm. The curve of A* Best Solution represents the computational time of finding the optimal path and A* complete illustrates the computational time of the A*-based global routing algorithm that exhaustively explores all the possible actions. From the comparison, it is obvious that the proposed GRTM algorithm outperforms the A*-based global routing algorithm in the computation complexity. Such a performance gap becomes apparent as the agent number increases. However, the A*-based global routing algorithm can quickly find an initial solution though the solution quality is far worse than the optimal one.

*2) The* `Ref-iSST` *algorithm:* The most significant difference between the `Ref-iSST` algorithm and `Bi-iSST` is that the heuristic value is estimated using the waypoints rather than the single target state in `Ref-iSST`. The waypoints are on the global reference trajectory generated by GRTM. Because the waypoints locate nearer to the current state, the heuristic value becomes more accurate. Therefore, the algorithm converges to the optimal solution much faster.

Algorithm 4 gives the basic structure of the `Ref-iSST` algorithm. In Line 1, we first generate a global reference trajectory $\mathcal{R}$, the cost of which underestimates the optimal solution. A new `Ref_Search_Selection` process (Line 4) is shown in Algorithm **??**. In the node selection process, we not only employ `iSST`-like search selection, but also keep a certain probability, $p$, to select vertices near the reference that prioritizes the propagation along the reference trajectory (Lines 5-6 in Algorithm **??**). Similar as the `SST` algorithm, the `Best_Near` process is used to select the least-cost vertex to the reference trajectory.

Algorithm 4 gives the basic structure of the `Ref-iSST` algorithm. The global reference trajectory $\mathcal{R}$ is generated under

**Algorithm 4:** Ref-iSST

1 $\mathcal{R} \leftarrow \texttt{GRTM}(\mathbf{q}_\texttt{s}, \mathbf{q}_\texttt{t})$
2 $\mathcal{G}, \mathbb{W} \leftarrow \texttt{init}(\mathbf{q}_\texttt{s}); \mathbb{E}, \mathbb{P} \leftarrow \emptyset$
3 **for** $m$ *iterations* **do**
4 $\quad x_\texttt{selected} \leftarrow \texttt{Ref\_Search\_Selection}(\mathcal{G}, \mathcal{R})$
5 $\quad \mathbb{X}_\texttt{temp} \leftarrow \texttt{Blossom}(\mathcal{R}, x_\texttt{selected}, l)$
6 $\quad$ **foreach** $x \in \mathbb{X}_\texttt{temp}$ **do**
7 $\quad\quad$ **if** not $\texttt{Collision\_Free}(\overline{x_\texttt{selected} \rightarrow x})$ **then**
8 $\quad\quad\quad$ **continue**
9 $\quad\quad$ **if** not $\texttt{Local\_Best\_SST}(x, \mathbb{W}, \delta_\texttt{s})$ **then**
10 $\quad\quad\quad$ **continue**
11 $\quad\quad \mathcal{G} \leftarrow \mathcal{G} \cup \{x\}$, $\mathbb{E} \leftarrow \mathbb{E} \cup \{\overline{x_\texttt{curr} \rightarrow x}\}$
12 $\quad\quad \texttt{Prune\_Dominated\_Nodes\_SST}(\mathcal{G}, \mathbb{W}, x)$
13 $\quad\quad$ **if** $\texttt{distance}(x, q_\texttt{t}) < \delta_\texttt{watch}$ **then**
14 $\quad\quad\quad \mathbb{P} \leftarrow \mathbb{P} \cup \{x\}$
15 $\pi \leftarrow \texttt{Get\_Best\_Trajectory}(\mathcal{G})$

an ideal model, so the cost underestimates the optimal solution. Not only does the propagating node selection process Ref_Search_Selection(Lines 4) employ iSST-like search selection, it also keeps a certain probability, $p$, to select vertices near the reference that prioritizes the propagation along the reference trajectory. The Blossom procedure (Line 5) generates a set $\mathbb{X}_\texttt{temp}$ containing $l$ temporary nodes by applying a series of propagation originated at $x_\texttt{selected}$ with random control inputs. The $l$ temporary nodes are prioritized according to the quality function shown in Eq. (36). Each temporary node is tested to avoid collision (Line 7) and keep sparsity (Line 9).

$$\texttt{quality}(x) = \frac{\exp(-k \cdot a \cdot \frac{|c - c_\texttt{max}|}{\max(c, c_\texttt{max})})}{\texttt{distance}(x, x_\texttt{ref})}, \tag{36}$$

where $c$ is the cost of the vertex $x$ and $c_\texttt{max}$ is the total cost of the reference trajectory. $x_\texttt{ref}$ is the estimated state in the reference trajectory corresponding to $c$ and the target state is selected as the reference state if the vertex's cost exceeds the total cost of the reference trajectory. Parameter $a$ represents the weight of cost in the quality measure and $k$ is a degrading factor with respect to $c$, where $k = 1$ for $c < c_\texttt{max}$ and $k > 1$ for $c > c_\texttt{max}$. The quality function aims to prioritize the vertex whose state is closer to the estimated state with the same cost in the reference trajectory. By setting $k > 1$ for $c > c_\texttt{max}$, the temporary state around the target state with less cost is prioritized.

## IV. Future Work

The motion controller works well for low speed scenario. However, when the speed reaches around 25 m/s in the simulation environment, the vehicle can't follow the reference trajectory, especially for the region with sharp turns. The reasons causing such failure may be:

1) the description of the dynamics model is not precise. Some degrees of freedom, like pitch and roll, are ignored in Eq.(1) and Eq.(2). And the calculation of the longitudinal friction force may not be correct in high speed. In order to improve the model, we have trained a recurrent neural network(RNN) model containing three dense layers and a activation layer to describe it. Each dense layer has 256 neural units. The RNN model can describe the dynamics precisely. However, when incorporating this model to MPC algorithm, the computation can't be finished within 0.1 second, which is the update frequency of the controller.
2) the computational speed can't catch up with the vehicle speed. Different from the traditional feedback controller that will give an output instantly, the output of NMPC has a time lag in recent PC configuration since it has to solve a nonlinear optimization problem. We set this time lag to 0.1 second. So the initial state $\boldsymbol{x}_0$ that has to be estimated in Eq.(34) by the Eq.(2) may not be precise enough.

The work of next stage is to modify the controller to increase the controllable speed. The current plan is:

1) linearizing the model. Linearizing nonlinear the dynamics model Eq.(2) near a fixed point. Some attempts has been taken by set this fixed point at $x_0$.
2) using a simpler curve to represent of the trajectory. Eq.(28) describes the centerline and the trajectory very well. However, The floor operator $\lfloor\rfloor$ is Non-base algebra operation, which is a huge computational cost in solving an OCP.

Since the local MPC doesn't need the whole trajectory information, we may adopt an curve that is combined of basic operations as the expression of the local segment of the reference trajectory.

For the motion planning, the time step of the beam reshaping process can't be set to big since the force due to the deformation at sharp turns is huge, causing the simulation is quite slow. We are going to develop an algorithm which is able to set an adaptive time step, to increase the simulation speed but guarantee the convergence.

## REFERENCES


[1] A. Eskandarian, C. Wu, and C. Sun, "Research advances and challenges of autonomous and connected ground vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 683–711, 2021.

[2] J. Li, J. Yi, Z. Liu, and J. Lu, "On the dynamic stability and agility of aggressive vehicle maneuvers: A pendulum-turn maneuver example," 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:110534693

[3] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," 2016.

[4] M. Meywerk, *Vehicle dynamics*. John Wiley & Sons, 2015.

[5] H. Pacejka and I. Besselink, *Tire and Vehicle Dynamics*. Elsevier Science, 2012.

[6] T. M. Inc., "Tire-road interaction," Natick, Massachusetts, United States, 2022. [Online]. Available: https://https://www.mathworks.com/help/sdl/ref/tireroadinteractionmagicformula.html

[7] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optim. Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, jul 2014.

[8] "Sampling-based robot motion planning: Towards realistic applications," *Computer Science Review*, vol. 1, no. 1, pp. 2–11, 2007.

[9] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Computer Science Dept. Oct.*, vol. 98, no. 11, 1998.

[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[11] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *Int. J. Robot. Res.*, vol. 35, no. 5, pp. 528–564, 2016.

[12] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *2011 IEEE Int. Conf. Robot. Autom.*, Shanghai, China, 2011, pp. 1478–1483.

[13] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," 2015.

[14] C. Truesdell and C. Truesdell, "Timoshenko's history of strength of materials (1953)," *An Idiot's Fugitive Essays on Science: Methods, Criticism, Training, Circumstances*, pp. 251–253, 1984.

[15] J. N. Reddy, *Introduction to the finite element method*. McGraw-Hill Education, 2019.

[16] D. J. Inman and A. N. Andry, "Some results on the nature of eigenvalues of discrete damped linear systems," *Int J Appl Mech*, vol. 47, pp. 927–930, 1980.

[17] M. Spagnuolo and U. Andreaus, "A targeted review on large deformations of planar elastic beams: extensibility, distributed loads, buckling and post-buckling," *Math. Mech. Solids*, vol. 24, no. 1, pp. 258–280, 2019.

[18] S. H. Kim, H.-S. Chang, C.-H. Shih, N. K. Uppalapati, U. Halder, G. Krishnan, P. G. Mehta, and M. Gazzola, "A physics-informed, vision-based method to reconstruct all deformation modes in slender bodies," *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 4810–4817, 2021.

[19] R. Levien, "The elastica: a mathematical history," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-103, Aug 2008.

[20] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-spline techniques*. Springer, 2002, vol. 6.

[21] G. M. Phillips and P. J. Taylor, *Theory and applications of numerical analysis*. Elsevier, 1996.

[22] R. Lot and F. Biral, "A curvilinear abscissa approach for the lap time optimization of racing vehicles," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 7559–7565, 2014.

[23] J. Filar and K. Vrieze, *Competitive Markov Decision Processes*. Springer New York, 2012. [Online]. Available: https://books.google.com/books?id=uXDjBwAAQBAJ