Proceedings of the 37th Chinese Control Conference
July 25-27, 2018, Wuhan, China

A Real-time Motion Planner for Intelligent Vehicles Based on A Fast SSTG Method

Junxiang Li[1], Chao Li[1], Zhenping Sun[1], Daxue Liu[1], Bin Dai[1], Xiaohui Li[1]

1. College of Intelligence Science and Technology,
National University of Defense Technology, Changsha, Hunan, 410073, P.R.China
E-mail: enginelee@yeah.net

Abstract: This paper proposes a motion planner for intelligent vehicles in real time. The motion planner can adaptively sample a number of states according to the initial state of the intelligent vehicle and road geometry. Then a novel method based on the state-space trajectory-generation method generates kinematically feasible trajectories which connected the initial state and the sampled states in a fast manner. At last, the final trajectories will be selected by a defined performance function. The experimental results demonstrate that proposed motion planner has an improvement in generating feasible trajectories. The average runtime of our method is approximately 5% of that of model-based predictive state-space trajectory-generation method. Moreover, our motion planner has the capability to deal with complex traffic environments in real time.

Key Words: motion planner, state-space trajectory generation, intelligent vehicles, spiral curve.

1 Introduction

It is widely believed that intelligent vehicles have great potential in improving driving safety and comfort. In the past two decades, technology on intelligent vehicles (IVs) has made remarkable breakthroughs. Motion planning is one of the core technologies for developing intelligent vehicles. A motion planner generates safe and feasible trajectories for intelligent vehicles, which meet the constraint of the environment and avoid collision with obstacles (e.g., other participant vehicles and pedestrians).

This paper proposes a real-time motion planner for IVs based on a novel method, which is called as the fast state-space trajectory-generation (SSTG) method. The motion planner can be separated into three modules within the planning cycle. 1) An adaptive sampling module, which is able to sample a number of possible states according to the initial state of the ego vehicle, as well as road geometry or a given reference path; 2) A trajectory generation module, which generates kinematically feasible trajectories from the initial state to the sampled states. Note that the presented module uses a novel method based on the state-space trajectory-generation method. Meanwhile, this module is the key element of our motion planner; 3) A performance evaluation module, which chooses the final trajectories through a defined performance function.

The novelty of this paper lies in two aspects. 1) One is that the proposed motion planner generated trajectories based on the geometric property of the spiral curves. The procedure avoids solving strong nonlinear equations of the constraints, which needs to be iteratively solved by a numerical integration method. Thus, the motion planner has a lower computational cost and faster runtime. 2) The other one is that the proposed motion planner considers not only the constraint of position and heading but also the constraint of curvature by using a curvilinear coordinate transformation. The generated trajectories are ensured to have continuous curvature and satisfy the constraints of the vehicle states. Thereby, the motion planner can generate kinematically feasible and curvature-continuous trajectories.

This work is supported by National Natural Science Foundation (NNSF) of China under Grant 61790565, 61751311, U1564214.

2 Related work

The trajectory-generation based motion planning generates a feasible trajectory set and chooses the optimal one from the set. The trajectory-generation based method can be divided into two categories [1] according to the different sampling ways: trajectory-generation method sampling in the control space (or control-space trajectory-generation method) and trajectory-generation method sampling in the state space (or state-space trajectory-generation method).

The control-space trajectory-generation method can generate trajectories which are easily tracked by a controller, but the trajectories may be too simple to deal with complex environment constraints. In [2], a control-space trajectory-generation algorithm based on arc tentacles is proposed, which generates a number of arcs with the different radius. In [3, 4], an algorithm using the Clothoid curve as trajectory primitive is presented taking into account the current curvature and the velocity of the IV.

The state-space trajectory-generation method can deal with complex environmental constraints because of the consideration of the environment, however, the method often needs large computing resource. The state-space trajectory-generation method always formulates the trajectory generation into a two-point boundary value problem (TPBVP). In [5–7], the steering control command of a carlike robot is expressed as a polynomial of the arc length. The unknown parameters of the polynomial are solved through the gradient-based shooting method. However, because of the nonlinearity of the vehicle model, the shooting method cannot be solved by analytical ways. It is likely to obtain no feasible solution or converge too slowly. Moreover, the iteration process requires high computing resource. Our previous work in [8, 9] designs a parameter lookup table to improve the convergence rate of the algorithm. Nevertheless, the inherent shortcomings of iteration process cannot be solved through the lookup table. Consequently, we propose our real-time motion planner based on a fast SSTG method in this paper.

The structure of this article is organized as follows. In Section 3, the novel method of our real-time motion planner

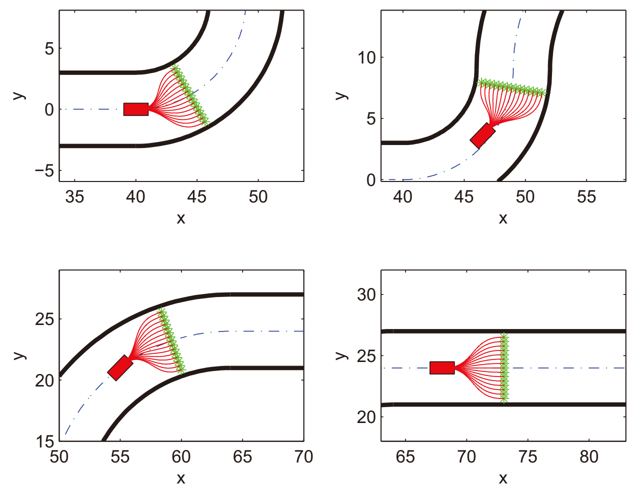
Fig. 1: Adaptive Sampling based on the state space.

is introduced specifically. In Section 4, the experimental results are described. Finally, conclusions and future work are discussed in Section 5.

3 Method

This section introduces details of proposed state-space trajectory-generation motion planner. According to the sequence of the planning cycle, our motion planner can be divided into three modules. 1) An adaptive sampling module, 2) A trajectory generation module and 3) A performance evaluation module. The basic function has been introduced in Sec. 1.

3.1 Adaptive sampling

Firstly, inspired by [1], we apply the adaptive sampling algorithm to sample the terminal states (i.e., position (x, y), headings θ, curvatures κ) adaptively according to the road geometry. In our method, the adaptive sampling parameters in structured environments consists of road centerlines X_{center} or drivable area S_{safe}, lane headings θ_{center}, lane curvature κ_{center}, lane width l_{width}, vehicle width v_{width}, lane horizons $d_{horizon}$, the number of lateral offsets to generate for each lane n_l, and the number of lanes n_p. Algorithm 1 generally operates by finding the state at some distance $d_{horizon}$ along each lane. In this particular example, the terminal positions are sampled with a uniform lateral offset δ, and calculated by a transformation matrix. The final vehicle angle and curvature are expected to be consistent with the centerline. As shown in Fig. 1, the sampling algorithm can adaptively generate the terminal states accustomed to road geometries.

3.2 Trajectory generation

After sampling, a feasible trajectory set is generated. All trajectories in the trajectory set are connected from the initial position to the sampling positions and complied with the heading and curvature constraints. Specifically, the raw trajectories are generated based on the symmetric configuration and spiral curves firstly without the curvature constraint of the states (both the initial state and the sampling states). Then, with consideration of the curvature constraint, trajectories will be regenerated based on the raw trajectories in a curvilinear coordinate system.

Algorithm 1 FINALSTATESAMPLING.

```
Require: $P_s$(sampling parameters), $X_I$(initial state)
Ensure: $X_S$(sampling terminal state)
 1: for $i = 0 \to n_p - 1$ do
 2:     for $j = 0 \to n_l - 1$ do
 3:         $X_{center} \leftarrow ComputeStates(S_{safe}, X_I, d_{horizon})$
 4:         $n \leftarrow i * n_l + j$
 5:         $\delta \leftarrow -0.5 * (l_{width} - v_{width}) + (l_{width} - v_{width}) * i/(n_p - 1)$
 6:         $\theta_{S,n} \leftarrow \theta_{center}$
 7:         $x_{S,n} \leftarrow x_{center} - \delta * sin(\theta_{S,n})$
 8:         $y_{S,n} \leftarrow y_{center} + \delta * cos(\theta_{S,n})$
 9:         $\kappa_{S,n} \leftarrow \kappa_{center}$
10:         $X_{S,n} = [x_{S,n}, y_{S,n}, \theta_{S,n}, \kappa_{S,n}]$
11:     end for
12: end for
```

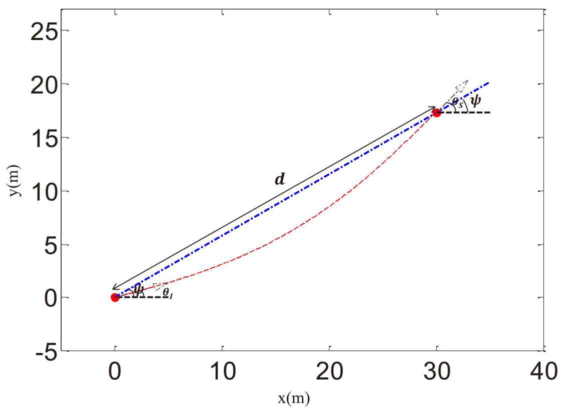

Fig. 2: Symmetric points in configuration space.

Symmetric configuration is a geometric property in the configuration space. The symmetric configuration refers to two points, whose average headings equals to the direction angle of the points.

$$\frac{\theta_I + \theta_S}{2} = \psi = \arctan(\frac{y_S - y_I}{x_S - x_I}) \tag{1}$$

When two points are symmetric configuration, they can be directly connected by a 'simple curve' (i.e., circular arcs and spiral curves), shown in Fig. 2. Even when two points are not symmetric configuration, they can still be connected through an intermediate point Q [10]. The intermediate point can be found as (2).

$$Q = \begin{cases} \{(x,y)|(x-x_I)(y-y_S) = (x-x_S)(y-y_I)\}, \\ \qquad\qquad \text{if } \theta_I = \theta_S \\ \{(x,y)|((x-x_I)(x-x_S)+(y-y_I)(y-y_S))* \\ tan(\frac{\theta_S-\theta_I}{2}) = (x-x_I)(y-y_S)-(x-x_S)(y-y_I)\}, \\ \qquad\qquad \text{if } \theta_I \neq \theta_S \end{cases} \tag{2}$$

The connection can be realized using a cubic spiral curve, whose heading is a cubic polynomial of the length of the curve. The cubic spiral curve can be calculated [10] as (3).

Table 1: The physical meaning of each cost term in cost function

Physical Meaning	Cost Term	Parameter Meaning
Cost for the length	$J_{s_i} = \frac{S_{max}-S_i}{S_{max}}$	S_i is the length of the path, S_{max} is the longest path in the planning period.
Cost for the smoothness	$J_{k_i} = \frac{1}{n}\Sigma_{i=1}^{n}\frac{\lvert\kappa_i\rvert}{\kappa_{max}}$	κ_i is the curvature of the extracted point of the path, κ_{max} is the maximal average curvature of all paths, n is the number of the extracted points of each path.
Cost for the consistence	$J_{d_i} = \frac{\lvert d_i^t - d_*^{t-1}\rvert}{d_{max}}$	d_i^t is the terminal deviation of the current path, d_*^{t-1} is the terminal deviation of the optimal path in the last planning cycle, $d_{max} = \max\limits_i^m \lvert d_i^t - d_*^{t-1}\rvert$.

As two points can be connected by the spiral curve, it is regarded as the raw trajectory.

$$\begin{aligned} \kappa(s) &= \frac{6\alpha D(\alpha)^3}{d^3} s(\frac{d}{D(\alpha)} - s), s \in [0, l] \\ D(\alpha) &= 2\int_0^{1/2} cos(\alpha(3/2 - 2t^2)t)dt \end{aligned} \tag{3}$$

where l denotes the total length of the curve, and d is the Euclidean distance between two points, α is the difference between two headings $\alpha = \theta_S - \theta_I$.

In order to comply with the curvature constraints of the initial state and the sampling states, we transform each raw trajectory to a curvilinear coordinate system. The curvilinear coordinate is obtained by the arc length $s(t)$ and perpendicular deviation $q(t)$ at a certain point of the center line [11]. The trick we use here is to exploit the raw trajectory as the center line. The perpendicular deviation $q(s)$ to the axis is expressed as a polynomial of arc length s. As the position and heading of endpoints of the raw trajectory are on the axis, the deviation is zero. The curvature constraint at the end of the trajectory κ_S can be supposed to be zero take into account the smoothness and the high replanning frequency. Therefore, with the curvature constraint, each trajectory in the curvilinear coordinate system can be represented as (4).

$$\begin{aligned} q(0) = 0, q(l) = 0 \\ \theta(0) = 0, \theta(l) = 0 \\ \kappa(0) = \kappa_0, \kappa(l) = 0 \end{aligned} \tag{4}$$

where l is the arc length of the curve, and κ_0 denotes the curvature of the initial state.

Considering that there are six constraints in (4), the order of the polynomial equation is determined to be five for a solvable solution.

$$q(s) = a_0 + a_1 s + a_2 s^2 + a_3 s^3 + a_4 s^4 + a_5 s^5 \tag{5}$$

In the curvilinear coordinate system, the heading and curvature comply with the following relationship [12].

$$\begin{aligned} tan(\theta) &= \frac{dq(s)}{ds} \\ \kappa(s) &= \frac{1}{Q}(\kappa_r + \frac{(1-q(s)\kappa_r)(d^2q/ds^2) + \kappa_r(dq/ds)^2}{Q^2}) \\ Q &= \sqrt{(dq/ds)^2 + (1-q(s)\kappa_r)^2} \end{aligned} \tag{6}$$

where κ_r is the curvature of the corresponding point in the raw trajectory. As the raw trajectory does not consider the curvature, $\kappa_b(0) = \kappa_b(l) = 0$.

Substitute (5) and (6) into (4), the parameters of the polynomial equation can be calculated as follows.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & l & l^2 & l^3 & l^4 & l^5 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2*l & 3*l^2 & 4*l^3 & 5*l^4 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6*l & 12*l^2 & 20*l^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \kappa_0 \\ 0 \end{bmatrix} \tag{7}$$

Then trajectories in the curvilinear coordinate system, which is denoted as $\Pi = (s, q, \theta(s), \kappa(s))$ are transformed to Cartesian coordinate system $(x', y', \theta', \kappa')$ (8). As previous mentioned, the generated trajectories satisfy the constraints of the initial state and the sampling state, including their curvature constraint.

$$\begin{aligned} x' = x_r - q * cos\theta_r \\ y' = y_r + q * sin\theta_r \end{aligned} \tag{8}$$

where x_r, y_r, θ_r is the position and heading of the corresponding point in the raw trajectory.

3.3 Collision-Test and Performance Evaluation

After generating paths, it is required to choose the optimal path to be followed. The choosing process works as follow. First, in order to ensure the safety, each path will be tested whether it will collide with obstacles. The method for collision test is using four incircles to approximate the shape of the vehicles [9, 12]. Points are extracted in the path, thus the inevitable collision can be regarded as the intersection between the list of incircles and the obstacles. Therefore, the distance between obstacles and the centers of circles are required to be greater than the circle radius for safety. If they are bound to collide with obstacles, they will be trimmed.

Additionally, all paths including the trimmed ones are evaluated by a cost function (9). The optimal path is the one that minimizes the cost function. The cost function is designed with consideration of following factors: 1) the length of the path after collision test, 2) the smoothness, which can be represented as the average curvature of the path, 3) the consistency of the path. Each factor will be modeled as a term in the function. The definition and the meaning of each factor are listed in Table 1.

$$i^* = \arg\min_{i=1}^{m}(w_s J_{s_i} + w_k J_{k_i} + w_d J_{d_i}) \tag{9}$$

where J_{s_i}, J_{k_i}, and J_{d_i} are costs, whose meaning are introduced in Table 1; w_s, w_k, and w_d are corresponding parameters; i is the index of path; m is the number of paths.

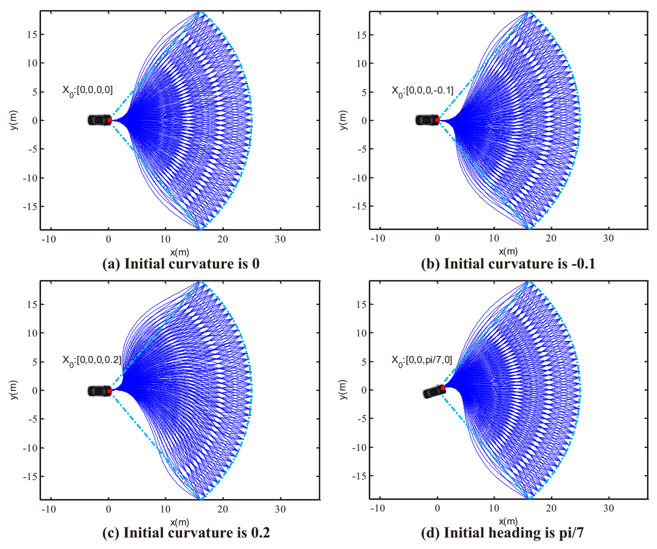

Fig. 3: Trajectory generation for different initial states.

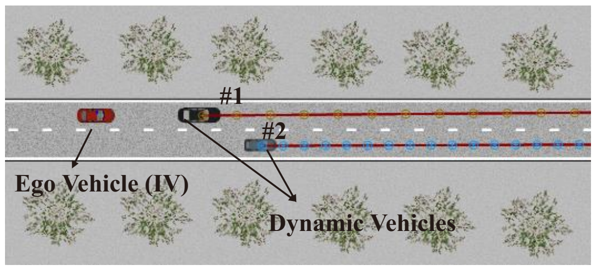

Fig. 4: Experimental scenario.

4 Experimental Results

In this section, we carry out two experiments. Firstly, a comparison of our novel trajectory generation method and method based on the model-based predictive method (MBP) [9] is discussed. Secondly, we conduct a simulation experiment in a typical driving environment to verify the effectiveness of our motion planning method. The platforms of our simulation experiments are Prescan 8.0 and Matlab R2012b. Prescan is a high-fidelity simulation environment for the development of intelligent vehicles. The parameters of simulation experiment are listed in Table 2.

Table 2: Main parameters in the simulation environment

Meaning	Value
Planning cycle	0.05 s
Lane Width	3.5 m
Vehicle Width	1.86 m
Vehicle Length	4.45 m

4.1 Comparison on different trajectory generation methods

In this section, we demonstrate a comparison between our fast SSTG method and the trajectory generation method based on MBP.

We selected four different initial states with different curvatures or headings. The terminal positions are distributed in a circular arc with the center of (0,0) and the radius of 25 meters. The sampling angles of the terminal positions in the arc are $[-\frac{5}{18}\pi, \frac{5}{18}\pi]$ rad, and the interval is $\frac{\pi}{72}$ rad. At each terminal position, the number of the sampling heading is five, and the resolution is $\frac{\pi}{18}$ rad. Note that the initial curvature selected only in three different values (i.e., 0, -0.1, 0.2 m^{-1}), the method can deal with larger initial curvature as well. The paths generated by the fast SSTG method are presented in Fig. 3. We can find that the trajectory will be generated in a smooth manner at different initial states.

The main metric that we use to compare the two methods are the number of feasible paths and the average runtime, which are listed in Table 3. As we can find in Table 3, in the same initial condition, the number of feasible trajectories utilizing our method is more than the one of MBP, while the time cost of our method is only about 5% of the one of MBP. As our method avoids numerical iteration, the results show a more notable real-time performance and better trajectory-generation capacity.

4.2 Performance of motion planner

In this section, a typical traffic scenario (shown in Fig. 4) is built to verify the effectiveness of our proposed motion planner. Due to the dynamic vehicles, the traffic scenario becomes complex as in reality. Our intelligent vehicle (i.e., ego vehicle) drives on a unidirectional two-lane road when there are two dynamic vehicles with different velocities in front. Dynamic vehicle #1 has a constant velocity of 15 m/s, and dynamic vehicle #2 has a constant velocity of 10 m/s. During the simulation experiment, vehicle #1 overtakes the vehicle #2.

The heading and the curvature of the planned trajectories generated by fast SSTP motion planner are shown in Fig. 5. The total recorded time is 20 seconds. The relationship of the curvature κ and the steering angle δ follows the bicycle model $\tan\delta = L\kappa$, where L is the wheelbase of the IV. The steering angle is defined as negative when it is on the right-hand side from its center and positive for the left-hand side. The maximal steering angle is 500^o. As shown in Fig. 5, our motion planner generates trajectories with continuous heading and curvature, thus guarantee the smoothness of the driving. The steering angle is positive relative to the curvature and is in the scope of $(-180^o, 180^o)$. The chosen screenshots at certain time (i.e., 1^{st}, 2^{nd}, 7^{th}, and 12^{th} second) of the experiments are shown in Fig. 6. The rough experimental procedure works as follows. At 1^{st} second, The intelligent vehicle changed lane to avoid closer dynamic vehicle #1 for the safety. Then, with dynamic vehicle #1 overtaking dynamic vehicle #2 (at 2^{nd} second), the IV gradually change back the left lane (at 7^{th} second). Though the behavior at 12^{th} second is not intuitive for the purpose, the whole driving is smooth and interactive.

5 Conclusions

The experimental results show that the presented motion planner (fast SSTP motion planner) greatly improves the real-time performance of trajectory generation and generates more possible trajectories in the trajectory set. The trajectories generated are not only adaptive to driving environments but also satisfying the constraint of the vehicle state. The

Table 3: The number of feasible trajectories and the average runtime using two trajectory generation methods.

Method	Initial Curvature(1/m)	feasible trajectory	average runtime(ms)
Ours	0	105	10.63
	0.1	105	10.62
	-0.2	105	10.52
MBP	0	102	206
	0.1	100	218
	-0.2	95	305

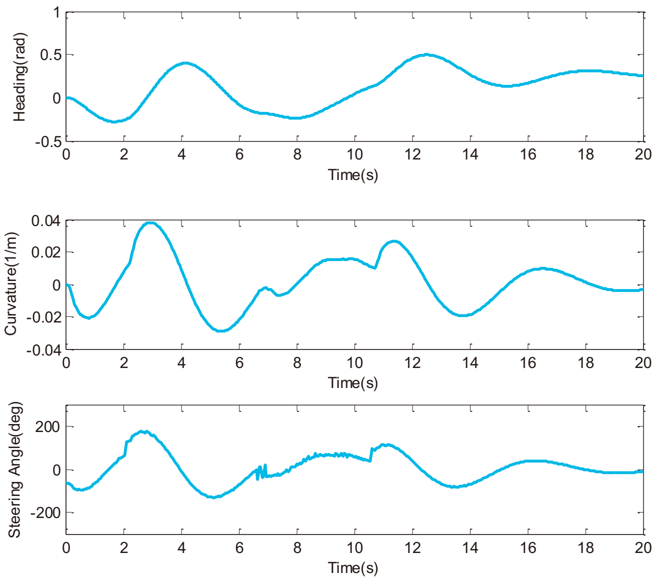

Fig. 5: Heading, curvature and steering angle.

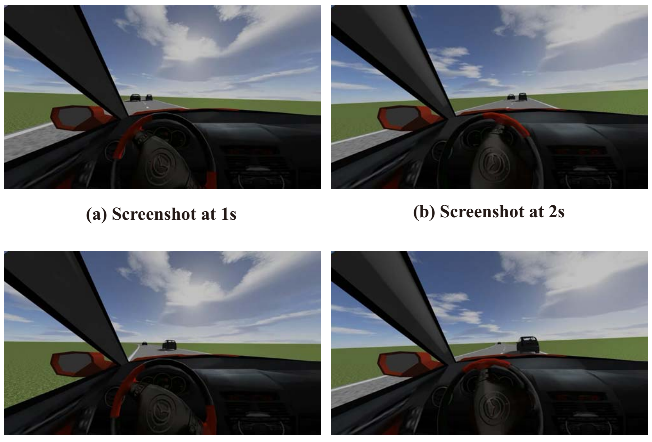

Fig. 6: Screenshots of the experiment.

driving using the fast SSTP motion planner is smooth and interactive with complex environments.

For the future work, we will design the evaluation performance function using learning methods to exploit the experience of the human drivers.

References

[1] T. M. Howard, C. J. Green, A. Kelly, and D. Ferguson, "State space sampling of feasible motions for highperformance mobile robot navigation in complex environments," *Journal of Field Robotics*, vol. 25, no. 67, pp. 325–345, 2008.

[2] F. von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller, and H.-J. Wuensche, "Driving with Tentacles: Integral Structures for Sensing and Motion," *Journal of Field Robotics*, vol. 25, no. 9, pp. 640–673, 2008.

[3] M. Himmelsbach, T. Luettel, F. Hecker, F. V. Hundelshausen, and H. J. Wuensche, "Autonomous off-road navigation for mucar-3 - improving the tentacles approach: Integral structures for sensing and motion," vol. 25, no. 2, pp. 145–149, 2011.

[4] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Shared steering control using safe envelopes for obstacle avoidance and vehicle stability," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 441–451, 2016.

[5] A. Kelly and B. Nagy, "Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 583–601, 2003.

[6] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.

[7] B. Nagy and A. Kelly, "Trajectory generation for car-like robots using cubic curvature polynomials," *Field and Service Robots*, vol. 11, 2001.

[8] X. Li, Z. Sun, D. Liu, Q. Zhu, and Z. Huang, "Combining local trajectory planning and tracking control for autonomous ground vehicles navigating along a reference path," *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 725–731, 2014.

[9] X. Li, Z. Sun, D. Cao, D. Liu, and H. He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles," *Mechanical Systems and Signal Processing*, vol. 87, Part B, pp. 118–137, 2015.

[10] B. I. Hartman, "Smooth Local-Path Planning for Autonomous Vehicles," pp. 263–284, 1991.

[11] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenét frame," *Proceedings - IEEE International Conference on Robotics and Automation*, no. June, pp. 987–993, 2010.

[12] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 2, pp. 740–753, 2016.