2019 IEEE Intelligent Transportation Systems Conference (ITSC)
Auckland, NZ, October 27-30, 2019

Path Planning for On-road Autonomous Driving with Concentrated Iterative Search

Qi Wang, Kuifeng Su and Dongpu Cao

***Abstract*— This work proposes a novel method of generating an on-road path comprising simple, short line segments, guaranteeing both dynamic and obstacle constraints with enough smoothness. Instead of certain forms of spline, simple line segments are directly used to form the path. Initially, a very raw path is generated consisting of several rough line segments in sequence. The initial path represents a rough guess that lies close to the optimal solution with relatively high possibility. Therefore, subsequent search shrinks its sample space to the neighborhood of the generated path to produce a finer path comprising shorter line segments. The same process is repeated until the line segment is shorter enough and becomes negligible to be followed by the ego vehicle. The proposed method only uses simple line segments, but the result naturally forms a very smooth path that meets the constraints needed in the end.**

***Index Terms*— Autonomous Driving, Path Planning, Concentrated Iterative Search**

I. Introduction

Autonomous driving technology has received considerable attention in recent years. However, path planning for on-road autonomous driving vehicles is still a challenging task as vehicles normally move in a relatively high speed. The path needs to be highly smoothed to ensure comfort and safety while maintaining enough flexibility to guarantee traversability along the road. Hence, intensive research has been carried out to come up with an ideal spline that can deal with this problem.

Generally speaking, there are two major categories of on-road path-planning algorithms for autonomous vehicles, namely sample-based and optimization-based algorithms. Sample-based algorithms construct a graph with motion primitives and then search for the best path in the graph. Some of earlier works [1][2] construct a path with arcs and lines. But these works only account for kinematic constraints with G^1 continuous and curvature discontinuous. This is not sufficient for vehicles with a higher speed and more flexible primitives are needed. Therefore, [3][4] composed trajectories from clothoids and lines with G^2 continuity. Later, [5] developed a numerical approach to solve the problem with an optimization-based technique, which was then applied by [6][7] to form the search graph along the given road. A closed-form solution [8] is also proposed to address the same problem with a couple of tunable parameters. Polynomials [9][10] are also used as motion primitives for sample-based motion planning. However, as mentioned in [10], the sample space cannot fully cover the entire solution space, and therefore, the solution of such algorithms can only be regarded as suboptimal.

Optimization-based approaches [11]–[13] are also widely used in path planning for autonomous driving due to its high-quality solutions that can be close enough to the optimum. However, the path-planning problem is mostly non-convex and contains multiple local minima. Unless the initial guess is close enough to the global minimum, it may take a long time to converge or end up with invalid results.

The combination of sample-based and optimization-based approaches [14][7][15] are also used so that each can counteract the weakness of the other. The sample-based path planning is used to find a initial guess in the neighborhood of the global optimum, while the optimization-based method is applied afterward to further approach the global optimum.

As mentioned above, multiple techniques must be combined to deal with on-road path planning. A novel sample-based on-road path-planning algorithm for autonomously driven vehicles will be proposed in the following sections. The proposed sample-based method can equally provide the global optimum without the help of any optimization-based method. Unlike the existing approaches that employ various splines, simple straight line segments are directly used as motion primitives in this work. This dramatically reduces the complexity and computation time of the on-road path planning task.

The proposed method divides a road into several sections, with each section containing a limited number of samples (Section II-A). A search graph is then created by linking the samples in the neighboring sections with simple line segments. Based on the cost function defined in Section II-D, a raw path is generated, leading the ego vehicle from its current state to the last section by searching the graph. The raw path is only a rough estimate of the solution. It has high probability of lying close to the optimal solution. Therefore, a subsequent search is applied (Section III) within a subspace around the neighborhood of the raw path. The subspace shrinks both laterally and longitudinally. The sample steps shrink accordingly, making the path gradually refined after several iterations of the search. Eventually, the search will converge to a smoothed non-parametric path consisting of a sequence of line segments of very short lengths.

As the sample space shrinks longitudinally in each iteration, there is another advantage— the search iteratively converges on the near range of the current location of the ego vehicle. This make the road section in the near range

Qi Wang and Kuifeng Su are with Autonomous Driving Lab, Tencent, China.
{kiwang, kuifengsu}@tencent.com
Dongpu Cao is with Department of Mechanical and Mechatronics Engineering, University of Waterloo, Canada.
dongpu.cao@uwaterloo.ca

more thoroughly evaluated than the road section far away. This is a common-sense feature as the path section in the near range is more urgent and has less variation than the road section in the future. Therefore, it makes no sense to put computation resources equally along all the road sections.

Note that the decoupling of speed-path planning [10][16][7] can dramatically reduce dimensions of the state space and subsequently reduces the complexity of the problem, without much loss of optimality. We take the same path-speed-decoupling method, assuming a speed plan has been made by other speed-planning task and focus our work purely on path planning.

II. Construction of Graph

The Frenet coordinate system has been widely used for the path planning of autonomous driving systems [9]. The transformation between Cartesian and Frenet frames depends on a reference line with a high order of continuity. It involves some extra cost that is required to smooth the provided raw reference line. Therefore, in this paper, we construct the search graph in the Cartesian frame. In fact, the concept of the proposed approach can be applied in the Frenet frame as well.

A. Samples and Links

In practice, the sample interval Δs of the road section would be affected by the speed plan $v(t)$, and we assume that the estimated speed is given by the speed planning task. However, for the sake of simplicity, the road between the ego vehicle and a limited horizon, $\overline{s}$, is sampled n^s sections with the same sample interval $\Delta s = 2\overline{s}/(n^s - 1)$. The sampled sections are labeled s_i sequentially. θ_i^r and κ_i^r are the orientation and curvature at s_i on the reference line, respectively.

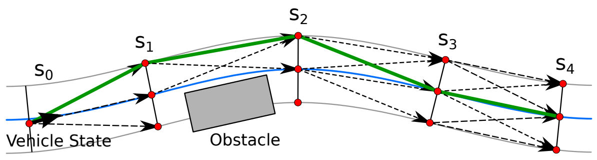


Fig. 1. Samples and links between neighboring sections.

As shown in Fig. 1, the central line of the target lane (shown as the blue line) is initially used as the reference line. Then n^l lateral positions l^k (shown as the red dot) are sampled at the same lateral step $\Delta l = 2\overline{l}/(n^l - 1)$ in each section s_i. The relative maximum lateral sample distance is $\overline{l}$. $\overline{l}$ depends on the free space of the vehicle along the road. There is only one lateral position l_0^0 in the first section s_0, which is at the same position as the ego vehicle.

The black dashed arrowed lines in Fig. 1 point to the next neighboring section. For any position l_i^k in section s_i, a link $\chi\{l_i^k \Rightarrow l_{i+1}^h\}$ will be created for each position l_{i+1}^h in the next section s_{i+1}. The creation of links should conform to the following rules.

First, there are links connecting l_i^k from one section earlier s_{i-1}. Second, the orientation of the link $\chi\{l_i^k \Rightarrow l_{i+1}^h\}$ should be within $(\theta_i^r - \overline{\theta}, \theta_i^r + \overline{\theta})$, where θ_i^r is the orientation of the reference line at s_i and $\overline{\theta}$ is the maximum relative orientation difference to θ_i^r. Third, the link should have no collision against obstacles.

For collision checking of the link, a similar approach in [10] is used. Based on the speed plan, the longitudinal location of the ego vehicle along the road at every time stamp can be estimated. Once there is an obstacle, it has an overlapped area with the ego vehicle along the longitudinal direction of the road at the same time point. Such obstacles will be marked on a proximity map. Both the collision checking and the calculation of the proximity cost (Section II-D) will be carried out on the proximity map.

B. Graph and Entries

The construction of the graph is shown in Alg. 1. A search graph consists of nodes $w(\theta, \kappa, J, w_p)$ connected by links, where w_p is the parent node of w in the graph and J is the cost of w (Section II-D).

Each l_i^k is attached to n_θ orientation entries, which cover the orientation space $\Theta_i = \{\theta | \theta_i^r - \overline{\theta} < \theta < \theta_i^r + \overline{\theta}\}$. Θ_i defines the orientation sample space centered on θ_i^r, and $\overline{\theta}$ defines the boundary of the sample space Θ_i. Each entry $W_i^k[\theta_j]$ saves the node w of the minimum cost in the search graph, which ever fell in the sub-area covered by $W_i^k[\theta_j]$. Note that the entry map W_0^0 of the lateral position l_0^0 in the first section s_0 only contains a single entry, which is initialized by the current state of the ego vehicle.

The construction of the graph starts from the first section s_0 in line 2, Alg. 1. For each updated entry w (line 6) of l_i^k, a new node w^{new} (Line 17) will be created for each link $\chi\{l_i^k \Rightarrow l_{i+1}^h\}$ connected to the next section s_{i+1}. Additionally, a new cost J^{new} will be calculated for this entry (Line 16). The entry w^{old} closest to θ (Line 11) will be selected (Line 19) from the entry map W_{i+1}^h of l_{i+1}^h, where l_{i+1}^h is one of the lateral positions in next section s_{i+1} linked by $\chi\{l_i^k \Rightarrow l_{i+1}^h\}$. If the old cost J^{old} of the current entry node w^{old} is less than the cost J^{new} of the new node w^{new} (Line 22), then w^{old} will be updated by w^{new} (Line 23).

After the entry nodes are generated section by section until the last section is reached (Line 3), the entries with the lowest cost of the last section will be considered as the last node of the optimal path. The path could be found by iteratively tracing back through its parent nodes.

C. Extension of Node

The location (x, y) of the sampled nodes is generated as described in Section II-A. However, the orientation θ and curvature κ of nodes are generated during the graph construction. As shown in Fig. 2, w_i and w_{i+1} are the subsequent nodes connected by link χ (shown as the black line). It is assumed that the vehicle translates between neighboring nodes w_i and w_{i+1} with constant curvature κ_{i+1}, shown by the dashed red curve in Fig. 2. It can be concluded that the angle $\Delta\theta_i$ between w_i and χ is the same as the angle $\Delta\theta_{i+1}$ between χ and w_{i+1}, and that there is $\Delta\theta_i = \Delta\theta_{i+1}$. Therefore, θ_{i+1} is calculated by (1). In general, the curvature

Algorithm 1 Graph Construction

```
1:  set cost of all entries in all entry map to $\infty$
2:  $s_i \leftarrow s_0$
3:  while $s_i$ is not the last section do
4:    for each $l_i^k$ in $s_i$ do
5:      get entry map $W_i^k$ of $l_i^k$
6:      for each updated entry $w$ in $W_i^k$ do
7:        $\theta^w \leftarrow$ orientation saved in $w$
8:        $\kappa^w \leftarrow$ curvature saved in $w$
9:        for each link $\chi$ of $l_i^k$ linked next section $s_{i+1}$ do
10:         $d \leftarrow$ length of $\chi$
11:         $\Delta\theta \leftarrow$ angle between $w$ and $\chi$
12:         $\theta \leftarrow \theta_w + 2\Delta\theta$
13:         $\kappa \leftarrow 2\Delta\theta/d$
14:         $\kappa' \leftarrow (\kappa - \kappa^w)/d$
15:         $J \leftarrow$ cost saved in $w$
16:         $J^{new} \leftarrow cost(\chi, \kappa, \kappa') + J$ (Section II-D)
17:         $w^{new} \leftarrow (\theta, \kappa, J^{new}, w)$
18:         get lateral position $l_{i+1}^h$ linked by $\chi$
19:         get entry map $W_{i+1}^h$ and subspace $\Omega_{i+1}$ of $l_{i+1}^h$
20:         find entry $w^{old}$ in $W_{i+1}^h$ closest to $\theta$
21:         $J^{old} \leftarrow$ cost saved in $w^{old}$
22:         if $J^{old} > J^{new}$ then
23:           update $w^{old}$ with $w^{new}$
24:   $s_i \leftarrow s_{i+1}$
```

of on-road path is relatively small, and the arc length l_{arc} is directly approximated by the length l_χ of χ, Thus, κ is approximated by (2). With the above method, a new extended node w_{i+1} in next section could be generated based on the current node w_i and link χ, as shown in Fig. 2. In the same manner, the approximate variation of the curvature κ'_{i+1} could be found by (3).

$$\theta_{i+1} = \theta_i + 2\Delta\theta_i \tag{1}$$

$$\kappa_{i+1} = \frac{2\Delta\theta_i}{l_{arc}} \approx \frac{2\Delta\theta_i}{l_\chi} \tag{2}$$

$$\kappa'_{i+1} \approx \frac{\kappa_{i+1} - \kappa_i}{l_\chi} \tag{3}$$

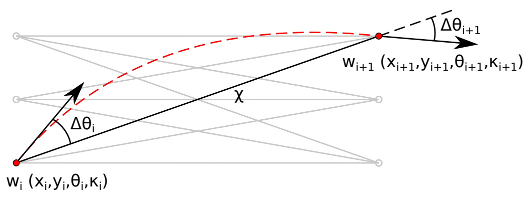


Fig. 2. Extension of node.

D. Cost Function

The cost function used in Alg. 1, Line 16 depends on κ, κ' and $\chi\{l_i^k \Rightarrow l_{i+1}^h\}$. It is the sum of four components defined by (4). While $\lambda_{lane}, \lambda_\kappa, \lambda_{\kappa'}$, and λ_{obs} are the weighted factors for each component:

$$J = \lambda_\kappa J_\kappa + \lambda_{\kappa'} J_{\kappa'} + \lambda_{lane} J_{lane} + \lambda_{obs} J_{obs} \tag{4}$$

J_κ is the curvature cost defined in (5)–this means minimization of the overall curvature along the path. κ' is the variation of the curvature along the path. $J_{\kappa'}$, defined in (6), is the cost that aims to minimize the variation of the curvature along the path. κ and κ' are calculated during the graph construction in Alg. 1, Lines 13, and 14.

$$J_\kappa = \kappa^2 \tag{5}$$

$$J_{\kappa'} = \kappa'^2 \tag{6}$$

Note that the calculation of J_{lane} and J_{obs} is carried out during the creation of links χ in Section II-A. J_{lane} is the cost of how far away the sample link χ is from the lane center. J_{lane} aims to keep the generated path close to lane center. The calculation of J_{lane} takes several samples along χ and sums up their lateral displacement L_i^{lane} relative to the lane center in (7), where d is the length of the current link χ.

$$J_{lane} = d\frac{\sum_{i=1}^{n} L_i^{lane}}{n} \tag{7}$$

J_{obs}, defined in (8), is the obstacle cost and represents how far away the path is from the obstacles. It aims to avoid the path getting too close to obstacles. Similarly, O_i^{obs} is the proximity distance of sample points along χ, which is measured on the proximity map. O^{thres} is the proximity threshold, with which the obstacle can be regarded safe and ignored by the ego vehicle.

$$J_{obs} = d\frac{\sum_{i=1}^{n}(O^{thres} - \min(O^{thres}, O_i^{obs}))}{n} \tag{8}$$

III. Concentrated Iterative Search

Alg. 2 shows the proposed method that iteratively refines the path P. The path P is initialized by the sampled reference line (Line 1). There are two nested loops. The outer loop in Line 3 is for path resampling (Section III-B). The inner loop in Line 4 is for path planning as well as sample space reduction (Section III-A). The inner loop searches the sampled space n_s times based on P, thereby ensuring that the sampled space is fully converged before P is resampled again in Line 10 and moves on to the next outer loop.

A. Reduction of Sample Space

When a path is created (the green line in Fig. 3), it is expected to lie in the neighborhood of the optimal solution. Therefore, later searches will be concentrated in the neighborhood of the resultant path by shrinking the sample space with $\eta < 1$ (Alg. 2 line 8). The subspace becomes a more focused search space to produce a better optimized solution, as shown by the pink line in Fig. 3.

Algorithm 2 Concentrated Iterative Search

```
$\bar{l}_{init}, \bar{\theta}_{init}$: initial value of $\bar{l}, \bar{\theta}$.
$P$: resulted path.
$S, X$: samples and links.
$G$: search graph.
 1: Initialize $P$ with the sampled reference line.
 2: Initialize $\bar{l}_{init}, \bar{\theta}_{init}$.
 3: for $n$ from 1 to $N$ do
 4:     for $m$ from 1 to $M$ do
 5:         $S, X \leftarrow generate\_samples\_links(P, \bar{l}, \bar{\theta})$
 6:         $G \leftarrow construct\_graph(S, X)$
 7:         $P \leftarrow search\_path(G)$
 8:         $\bar{l}, \bar{\theta} \leftarrow \eta\bar{l}, \eta\bar{\theta}$
 9:     if $n \neq N$ then
10:         $resample\_path(P)$
11:         $\bar{l}_{init} \leftarrow \bar{l}_{init}/2$
12:         $\bar{l}_{init}, \bar{\theta}_{init} \leftarrow \gamma\bar{l}_{init}, \gamma\bar{\theta}_{init}$
13:         $\bar{l}, \bar{\theta} \leftarrow \bar{l}_{init}, \bar{\theta}_{init}$
14: return $P$
```

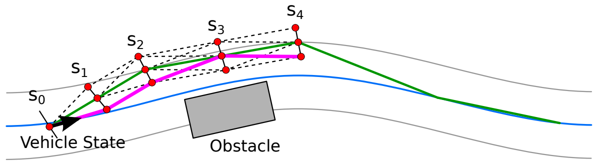


Fig. 3. Based on the path (shown in green lines) generated in previous iteration, the result of the subsequent search is shown in pink lines.

B. Resampling of path

For each outer loop (Alg. 2 in line 3), the resultant path is resampled in Line 10. For each pair of neighboring nodes (shown as red nodes in Fig 4), a new node (green nodes) will be inserted in between. The location of the new node w^* could be calculated by interpolating the midpoint along the arc connecting the neighboring nodes shown in Fig. 4. The resampled path is indicated by the green line. θ and κ of each node along the resampled path are recalculated, as described in Section II-C.

Additionally, since the sample interval along path shrinks to approximately half its original size, the lateral sample limit shrinks accordingly with the same magnitude (Alg. 2 in Line 11) to maintain the sample orientation interval at the same size.

There will be less lateral or orientation variation along a shrunk sampled path interval. Therefore, the initial sample limit $(\bar{l}_{init}, \bar{\theta}_{init})$ for each inner loop is supposed to shrink slightly with $\gamma < 1$ (Alg. 2 in line 12). It turned out that a smaller γ helps to produce a smoother result with less inner loops. However, a very small value of γ may leave much less space for obstacle avoidance for the following iterations and could lead to failure.

Finally, a highly smooth path in short line segments could be generated by iteratively shrinking the sample space.

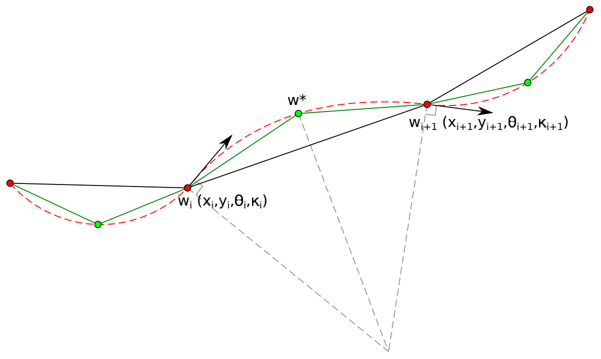

w*
w_{i+1} $(x_{i+1}, y_{i+1}, \theta_{i+1}, \kappa_{i+1})$
w_i $(x_i, y_i, \theta_i, \kappa_i)$

Fig. 4. Resample of path

IV. Experiments and Evaluation

Some experiments are done in this section to show the performance of the proposed approach.

A. Result of Subsequent Iterations

This section shows the variation of the result in subsequent iterations (the outer loop of Alg. 2 in Line 3). As shown in Fig. 5, the red rectangular objects represent the locations of obstacles. After each iteration, the resultant path is resampled, and a subsequent search is carried out based on the resampled path. This result shows how the path is iteratively refined and becomes smoother.

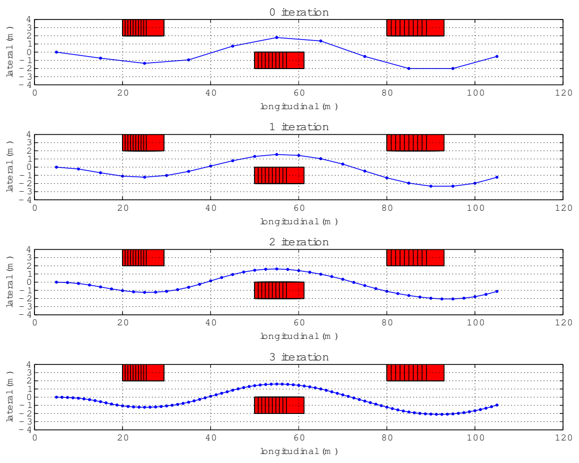
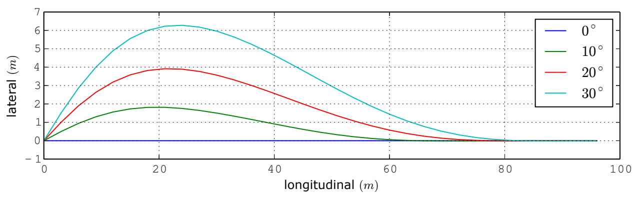


Fig. 5. Result after four iterations

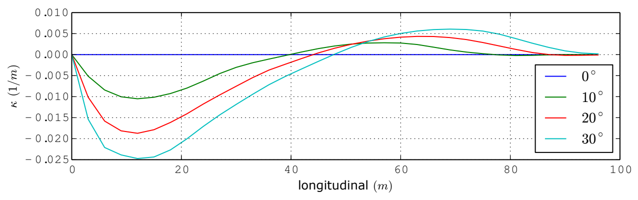
B. Initial State

In this section, the results of different initial conditions are compared. Fig. 6(a) shows the resultant path of different initial orientations–namely 0°, 10°, 20°, and 30°. Fig. 6(b) shows the corresponding variation of the curvature along the path shown in Fig. 6(a). Fig. 7(a) provides several examples of different initial curvatures–0, 0.01, 0.02, and 0.03 $1/m$, respectively. Fig. 7(b) shows how the curvature varies along each path in Fig. 7(a).

Fig. 6 and Fig. 7 show that the paths with different initial conditions can gradually steer back and get stabilized at the lane center. The result demonstrates that the proposed approach can tackle the starting state of different orientations and curvatures.

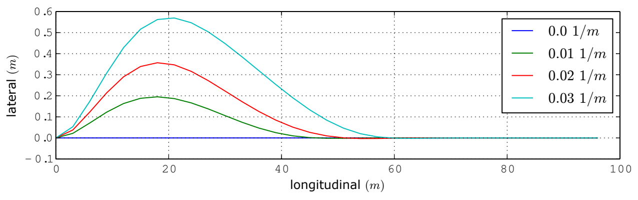


(a)

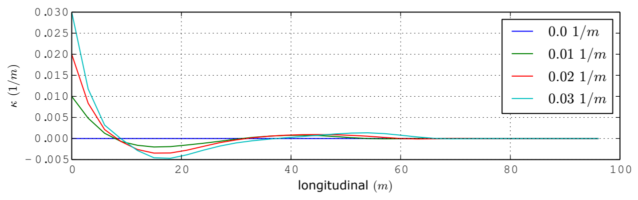


(b)

Fig. 6. Results of different initial orientations: (a) path, (b) curvature



(a)



(b)

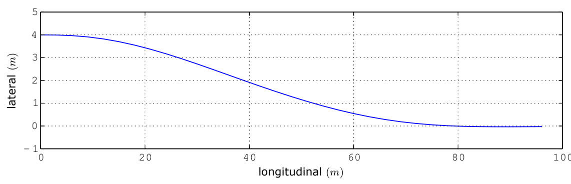
Fig. 7. Results of different initial curvatures: (a) path, (b) curvature

C. Lane change

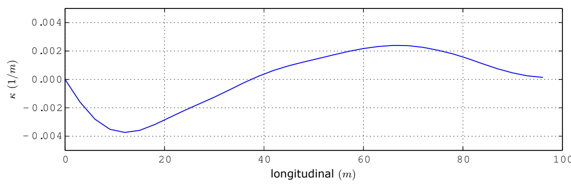
This experiment shows the result of a lane change. As shown in Fig. 8(a), the proposed approach produces a very smooth lane-change path. Fig. 8(b) illustrates the variation of the curvature along the path. It can be observed from Fig. 8(b) how smoothly the curvature varied during the lane-changing process.

D. Result in Dynamic Environment

Fig. 9 shows a simulation of on-road dynamic environment. The red rectangle represents the location of the ego vehicle moving in the lane-keeping state. The road is



(a)



(b)

Fig. 8. Lane change: (a) path, (b) curvature

comprised of three lanes. The dotted red line is the planning result of the proposed approach. As shown in Fig. 9(a), there is a blue vehicle moving on the left lane. In Fig. 9(b) and Fig. 9(c), the blue vehicle made a sudden lane change to the current lane. To avoid the collision, the proposed approach produces a path and let the ego vehicle flexibly steer over and smoothly return to the lane center (Fig. 9(d)). This result shows a very smooth and flexible performance of the proposed approach in a dynamic environment.

V. Conclusion

To sum up, the proposed approach offers some excellent features. To begin with, it can be directly applied in the Cartesian frame, making the implementation of the kinematic and dynamic constraints more straightforward. Second, it does not depend on a high-order continuous reference line, thus reducing the need for a smooth reference line. In other words, the proposed approach can tolerate a short range of discontinuity along the reference line. Third, the approaches that use certain types of spline always present additional constraints. Simple line segments are used to form a smooth path. This provides the path with more flexibility, dramatically reducing the complexity and computation cost. Fourth, unlike the optimization-based method which may be trapped in the local minimum, the proposed approach always converges to the global optimal solution. Finally, the proposed approach treats different longitudinal ranges of the road section with different computation costs. This coincides with common sense and saves computation resources.

References

[1] L. E. Dubins, “On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents,” *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[2] J. A. Reeds and R. A Shepp, “Optimal paths for a car that goes both forwards and backwards,” *Pacific Journal of Mathematics*, vol. 145, no. 2, 1990.

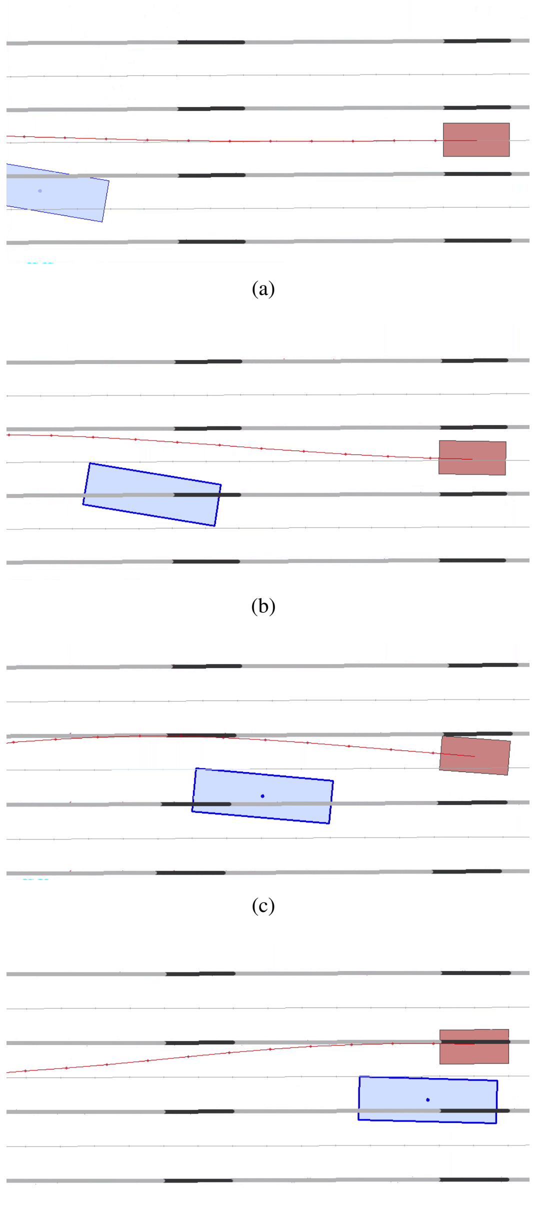


Fig. 9. Results in the dynamic environment

[3] D. Shin, S. Singh, and W. Whittaker, “Path Generation for a Robot Vehicle Using Composite Clothoid Segments,” in *IFAC Proceedings Volumes*, vol. 25, 1992, pp. 443–448.

[4] T. Fraichard, A. Scheuer, T. Fraichard, and A. Scheuer, “From Reeds and Shepp ’ s to continuous-curvature paths,” vol. 20, no. 6, pp. 1025–1035, 2005.

[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] M. Mcnaughton, C. Urmson, J. M. Dolan, and J.-w. Lee, “Motion Planning for Autonomous Driving with a Conformal Spatiotemporal Lattice,” in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4889–4895.

[7] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, “A real-time motion planner with trajectory optimization for autonomous vehicles,” in *IEEE International Conference on Robotics and Automation*, 2012, pp. 2061–2067.

[8] A. Piazzi, C. Guarino Lo Bianco, and M. Romano, “Smooth Path Generation for Wheeled Mobile Robots Using η 3-Splines,” *IEEE Transactions on Robotics*, 2007.

[9] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, “Optimal trajectories for time-critical street scenarios using discretized terminal manifolds,” *International Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, 2012.

[10] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, “Baidu apollo em motion planner,” 2018.

[11] B. Gutjahr, L. Gröll, and M. Werling, “Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1586–1595, 2017.

[12] J. Ziegler, P. Bender, T. Dang, and C. Stiller, “Trajectory planning for Bertha - A local, continuous method,” in *IEEE Intelligent Vehicles Symposium*, 2014, pp. 450–457.

[13] X. Qian, F. Altch, P. Bender, C. Stiller, and A. D. L. Fortelle, “Optimal Trajectory Planning for Autonomous Driving Integrating Logical Constraints : A MIQP Perspective,” 2016.

[14] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Path planning for autonomous vehicles in unknown semi-structured environments,” *International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[15] W. Lim, S. Lee, M. Sunwoo, and K. Jo, “Hierarchical Trajectory Planning of an Autonomous Car Based on the Integration of a Sampling and an Optimization Method,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 613–626, 2018.

[16] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J. W. Lee, “Tunable and stable real-time trajectory planning for urban autonomous driving,” in *IEEE International Conference on Intelligent Robots and Systems*, 2015, pp. 250–256.