2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Decoupled Sampling-Based Velocity Tuning and Motion Planning Method for Multiple Autonomous Vehicles

Fatemeh Mohseni[1] and Lars Nielsen[2]

Abstract— **This paper describes a decoupled sampling-based motion-planning method, based on the rapidly-exploring random tree (RRT) approach, that is applicable to autonomous vehicles, in order to perform different traffic maneuvers. This is a two-step motion-planning method including path-planning and motion timing steps, where both steps are sampling-based. In the path-planning part, an improved RRT method is defined that increases the smoothness of the path and decreases the computational time of the RRT method; it is called smooth RRT, SRRT. While some other RRT-based methods such as RRT* can perform better in winding roads, in the problem of interest in this paper (which is performing some regular traffic maneuvers in usual urban roads and highways where the passage is not too winding), SRRT is more efficient since the computational time is less than for the other considered methods. In the motion timing or velocity-tuning step (VTS), a sampling-based method is introduced that guarantees collision avoidance between different vehicles. The proposed motion-timing algorithm can be very useful for collision avoidance and can be used with any other path-planning method. Simulation results show that because of the probabilistic property of the SRRT and VTS algorithms, together with the decoupling feature of the method, the algorithm works well for different traffic maneuvers.**

I. INTRODUCTION

Motion planning plays an important role in the navigation of autonomous vehicles. It guarantees that a trajectory from the initial point to the goal point will be found, in the presence of constraints, such as collision avoidance, speed limits, and rules of motion.

A vast introduction to motion and path-planning problems, existing techniques, and solutions can be found in [1], [2]. For an ideal motion planner, a few requirements exist, including computational complexity, optimality, and completeness. However, only a few techniques attempt and are able to solve a planning problem in its complete generality, see [3].

Several heuristic search algorithms for path-planning have been proposed and used in a known workspace, such as A*, RRT*, and D*, [4], [5], [6]. There are some other techniques such as the methods that are based on trajectory-planning Model Predictive Control (MPC), which enable the planning and re-planning of trajectories, while taking into account the complex vehicle dynamics [7]. For example, an algorithm based on MPC is proposed in [8] for real-time obstacle avoidance for ground vehicles. MPC has also been combined with motion primitives in [9] in order to plan controls for the fast maneuvering of ground vehicles. For autonomous vehicle applications, where the vehicle has to move in an obstacle-rich environment, the computational complexity of the motion-planning algorithm is an important aspect. Since the vehicles usually move at high speed, the path planner has to find a collision-free path quickly.

The authors are part of the Department of Electrical Engineering, Linköping University, SE581-83, Linköping, Sweden. [1]fatemeh.mohseni@liu.se, [2]lars.nielsen@liu.se

In this work, we develop a two-step sampling-based motion-planning approach for autonomous vehicles with nonholonomic dynamics. There are other two-step motion-planning methods in the literature, e.g., [10], but the differences between those methods and this work are that in our method, both steps are sampling-based with nonholonomic dynamics and the first step has a different definition of parents in order to decrease the computational time and shorten the path length, while in the other works none of the steps or in some only one part is sampling-based with the common definition of parents and for simpler dynamic models, see, e.g., [1]. In our method, the path will be found in the first step, and then, the second algorithm tries to maximize the speed of each vehicle as much as possible, while taking into account the speed and acceleration limits. This motion-timing algorithm can be very useful for collision avoidance and can be used with any other path-planning method. By using the path-planning step first, and then the motion-timing step for collision avoidance, the computational time of the planner is reduced compared with similar existing approaches that control multiple vehicles.

In many scientific researches, probabilistic sampling-based methods, such as the rapidly-exploring random trees algorithm (RRT), [11], the probabilistic roadmap algorithm (PRM), [12], RRT*, [6], and the PRM* have been developed for robot and vehicle path-planning. Optimal RRT (RRT*) has been widely used in motion-planning for robots and vehicles. This algorithm is a modified version of RRG, see [13]. In RRT* the edges that are not part of the shortest path and are called redundant edges are removed in order to avoid the formation of cycles. Although these sampling-based algorithms are not complete, they provide probabilistic completeness to ensure that planning is as successful as possible. As it has been shown in [5], the choice of a distance metric as a cost function to find the nearest node affects the performance of RRT-based algorithms significantly. In PRM, the connections are attempted between road-map nodes that are within a fixed radius from one another, see, e.g., [14]. In both methods, the definition of nearest point in RRT has been changed, while in this work we propose the SRRT path finder algorithm, in which the definition of parents in RRT has been modified, in order to obtain smoother paths.

The RRG algorithm is another RRT-based method in which every time a new point is added, the connections are attempted from all other nodes in the graph that are within a ball of a non-fixed radius, see [13]. But, in this work, in order to increase the smoothness, make the path as short as possible, and reduce the computational time, we developed the SRRT algorithm which is appropriate for the traffic scenarios that are of interest in this paper. In this method, instead of finding one nearest point, a $K-$nearest vector is defined.

In this paper, the task for control of the vehicles is to

find trajectories that connect the initial and destination points without causing a collision in the presence of static and dynamical obstacles, and to meet the dynamical constraints. For our motion-planning problem, we introduce two RRT-based algorithms. The main reasons for choosing RRT as the base method are: 1) sampling-based algorithms can be applied to very general dynamical models; 2) the incremental nature of RRT lends itself easily to online implementation, while it guarantees retaining certain completeness; and 3) RRT does not require the explicit expression of constraints, but allows trajectory-wise checking of possibly very complex and varying constraints.

II. Preliminaries and method overview

In this section, the motion objective, system dynamics, and constraints are defined. The aim is to control multiple autonomous vehicles on their journey toward desired destinations in a decoupled way using sampling-based planning methods.

A. Motion-Planning Method

In this paper, the multiple-vehicle motion-planning problem is addressed. The task of each vehicle is to arrive at the destination point in the presence of obstacles, while meeting the dynamical limitations and avoiding inter-vehicle and vehicle/obstacle collisions. In order to reduce the complexity of multi-vehicle motion-planning, the problem is decoupled into path-planning and motion timing (velocity-tuning) parts; for each, a sampling-based method is introduced.

In the path-planning part, the paths of the vehicles are designed, while ignoring vehicle–vehicle interactions. In this step, an improved RRT (SRRT) method, defined in Section III-A, is proposed, which increases the smoothness and shortens the path and decreases the computational time of the RRT-based methods. This algorithm tries to achieve this by modifying the definition of parents and biasing.

In the SRRT method, after choosing a random point, the nearest point in the tree to that random point is found. Then, the K-nearest vector is defined, which includes the nearest point in the tree and $K-1$ parents of it. Between all the nodes of this vector, the one that does not collide with the static obstacles and has the least cost will be chosen as the nearest point. The cost is defined as the summation of the total traveled distance and the length of the direct path from the current point to the destination.

In the motion-timing step, which is a prioritized planning stage, a sampling method is proposed that guarantees the collision avoidance between different vehicles and tries to minimize the maneuver time as much as possible. The vehicles are sorted by priority and the motion for the higher-priority vehicle is planned first. After that, the motions of lower-priority vehicles are planned by considering the higher priority vehicles as moving obstacles.

The configuration space, see, e.g., [1], for the other vehicles is built in a $T \times S \in [0, t_f] \times [0,1]$ space. T indicates the time, S is the portion of the path that has been found in the path-planning step, and t_f shows the maximum maneuver time. In this space, any (t,s) point, $t \in T,\ s \in S$, that collides with higher-priority vehicles is shown as an static obstacle point for lower-priority vehicles.

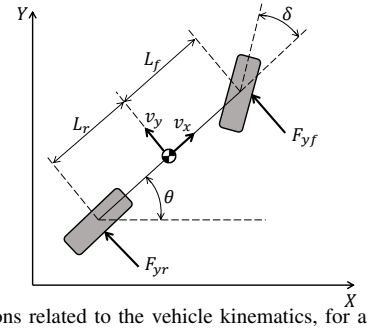


Fig. 1: Notations related to the vehicle kinematics, for a bicycle model.

After building the configuration space for each vehicle, the velocity-tuning sampling-based (VTS) method is used. At each step of this algorithm, a few random points are chosen on the $s = 1$ line and a number of random points are chosen in front of the vehicle, in an allowed interval, using a shooting method. Between all random points at each step, the one with the least distance to $s = 1$, which satisfies acceleration and wheel forces constraints, and does not collide in the $T - S$ space is chosen as the nearest point. The minimum distance is used as a criterion in order to minimize the maneuver time. By integration of the trajectory that is found using this algorithm and the dynamics of the vehicle, the velocity of the vehicle at each time instant can be calculated. This algorithm is an important idea that can be applied elsewhere in multiple vehicle systems and with other path-planning methods.

B. Vehicle Dynamics

It is assumed that the movement of each vehicle is described by a simplified bicycle model, in order to reduce complexity and computational effort, see [15]. The model is given by

$$\begin{cases} \dot{X} = v_x \cos(\theta) - v_y \sin(\theta) \\ \dot{Y} = v_x \sin(\theta) + v_y \cos(\theta) \\ \dot{\theta} = r \\ \dot{v}_x = \frac{-F_{yf}}{m} \sin(\delta) + v_y r \\ \dot{v}_y = \frac{F_{yf}}{m} \cos(\delta) + \frac{F_{yr}}{m} - v_x r \\ \dot{r} = \frac{L_f}{I_z} F_{yf} \cos(\delta) - \frac{L_r}{I_z} F_{yr} \end{cases} \tag{1}$$

where $p = (X,Y)$ is the Cartesian coordinates of the vehicle's center, θ is the orientation angle, v_x and v_y are longitudinal and lateral speeds, respectively, r is the yaw rate, δ is the steer angle, L_f and L_r are the distances from the center of gravity to the front and rear wheel, respectively, and I_z is the mass moment of inertia about the z-axis (see Fig. 1). For simplicity, aerodynamic resistance is neglected, and therefore, the longitudinal tire force becomes zero. In addition, F_{yf} and F_{yr}, which are the lateral forces acting on the front and rear wheel, are given by

$$\begin{cases} F_{yf} = -C_{\alpha f} \cdot \alpha_f \\ F_{yr} = -C_{\alpha r} \cdot \alpha_r \end{cases}, \quad \begin{cases} \alpha_f = (v_y + L_f r)/v_x - \delta \\ \alpha_r = (v_y - L_f r)/v_y \end{cases} \tag{2}$$

where α_f and α_r are the slip angles of the front and rear wheels, respectively. For more details about vehicle dynamics, the reader is referred to [16]. In addition to the vehicle dynamics described by the differential equations (1), constraints on steer angle and speed should also be considered to reflect the physical and mechanical limitations. These constraints are as follows.

$$v_{\min} \le v_x(t) \le v_{\max}, \qquad |\delta(t)| \le \delta_{\max} \tag{3}$$

III. Decoupled sampling-based planning (DSBP) method for multiple vehicle motion

In this section, the motion-planning problem for multiple vehicles is addressed. The task of each vehicle is to arrive at its destination point in the presence of static obstacles and meet the dynamical constraints, while avoiding inter-vehicle and vehicle/obstacle collisions.

In our proposed decoupled method, the problem is divided into two parts. First, the path-planning for each vehicle is performed by using the SRRT method, see Section III-A. Second, the motion-timing part is achieved by using a semi-deterministic sampling-based method that guarantees collision avoidance between different vehicles, see Section III-B. The total algorithm is thus a decoupled sampling-based velocity tuning and motion planning algorithm. For shortness, it is named DSBP (Decoupled Sampling-based Planning).

A. SRRT algorithm for path-planning

In the SRRT algorithm, the definition of parents in standard RRT, see [1], is modified in order to get a smoother path and also to increase the probability of using each random point. In addition, in order to increase the convergence rate, a biasing approach that is common in many sampling-based methods, see, e.g., [1], has been added in the SRRT algorithm. In the biasing part, while the road map has not been constructed, a random value β will be generated. If this random value is larger than the specified probability threshold β', then the random point, p_{rand}, will be generated randomly inside the terminal zone; otherwise, it will be generated in the whole space. The terminal zone is a circle centered at the desired destination point with a radius of a. This biasing method is used to increase the convergence of the method toward the destination point. After this step, the nearest node in the tree, p_{near}, from p_{rand} is found. By using p_{near} and p_{rand}, the K-nearest vector, P_{near}, is found according to Definition 1.

Definition 1 (K-nearest vector): K-nearest vector, $P_{\text{near}} = (p_{\text{near},1}, \ldots, p_{\text{near},K})$, for each random point is defined as its nearest point in the tree, the parent of that nearest point, and the $K-2$ parents of that parent.

The control inputs in SRRT step for dynamical model (1) are $v_x(t)$ and $\delta(t)$ where the longitudinal speed v_x is considered to be a constant value. After finding P_{near}, the proper steer angle to move from each component of that in the direction of the sample point is calculated and fed into the vehicle dynamics. By using the steer angles, the integration takes place for a given time horizon, which results in $P_{\text{new}} = (p_{\text{new},1}, \ldots, p_{\text{new},K})$. In this paper, the integration of the system is done using a numerical fourth order Runge-Kutta integration technique. For all the integrated points $P_{\text{new}}(i)$, a cost is calculated according to the following definition.

Definition 2 (Cost Function): For each component i of $P_{\text{new}} = (p_{\text{new},1}, \ldots, p_{\text{new},K})$, the cost function is defined as follows:

$$F(i) = G(p_{\text{start}}, P_{\text{new}}(i)) + H(P_{\text{new}}(i), p_{\text{goal}}) \tag{4}$$

where p_{start} and p_{goal} indicate the start and destination points, respectively, and G is the traveled distance from the start point to $P_{\text{new}}(i)$. Moreover, for each integrated point $p_{\text{new},i}$, if there is a collision between the path from $p_{\text{near},i}$ to $p_{\text{new},i}$ and obstacles, the heuristic cost $H(i)$ is set to infinity. Otherwise, $H(i)$ is the direct distance from $p_{\text{new},i}$ to the destination point.

The next step is to select the best element of P_{new}, p^*_{new}, which has the minimum cost value. The road map will be constructed when p^*_{new} arrives at the terminal zone. The SRRT algorithm is shown in Algorithm 1.

Algorithm 1 SRRT Algorithm

```
function PATH FINDER
    Tree-init{𝒢 = (𝒱, ℰ), E = ∅, 𝒱 ← p_start, p_goal,
    β' ∈ [0,1], K > 1, a > 0}
    while !flag do
        Generate a β ∈ [0,1]
        if β ≥ β' then
            p_rand = (p_goal(1) + a + 2.a.β, p_goal(2) − a +
             2.a.β)^T
        else
            p_rand ← SamplePoint()
        end if
        p_near = Nearest(𝒢(𝒱, ℰ), p_rand)
        P_near ← K-near(p_near, K)
        P_new ← Steer(P_near, p_rand)
        for i = 1 to K do
            if !Collision-free(P_new(i), P_near(i)) then
                H(i) = ∞
            else
                H(i) ←Directpath(P_new(i), p_goal)
            end if
            F(i) = G(i) + H(i) according to Definition 2
        end for
        n = arg min_i F(i)
        p*_near = P_near(n)
        p*_new = P_new(n)
        𝒱 ← 𝒱 ∪ {p*_new}
        if ‖p*_new − p_goal‖ ≤ a then
            return flag = True
        else
            return flag = False
        end if
    end while
end function
```

B. Motion Timing (Velocity Tuning) Sampling-Based Method (VTS)

Majority of the notations in the Velocity Tuning part are adopted from [1]. In the motion-timing part, a timing function, $\sigma : T \to [0,1]$, is designed for each vehicle, based on the priority. It has been assumed that some collision-free path τ has been computed already using the SRRT algorithm or any other path-planning method. σ indicates the location of the vehicle along the path τ at time t. $Z = T \times S$ defines the state space in which the time $t \in T$ and the position along the path, $s \in [0,1]$, are indicated by a point (t,s). The obstacle region in Z is defined as

$$Z_{\text{obs}} = \{(t,s) \in Z | \tau(s) \cap \mathscr{O}(t) \neq \emptyset\} \tag{5}$$

where $\mathscr{O}(t)$ indicates the obstacle region. Therefore, Z_{free} is defined as $Z_{\text{free}} = Z \setminus Z_{\text{obs}}$.

By defining the composition $\phi = \tau \circ \sigma$, which maps from T to Z_{free} via [0, 1], σ is achieved. Therefore, $\phi(t) = \tau(\sigma(t))$ indicates the configuration at time $t \in T$. We assume that the vehicle with the highest priority moves along its path, which was found in the first step, with any speed profile. For the other vehicles, after finding the path using the SRRT algorithm, the configuration $\phi(t)$ should be calculated. Each vehicle should move along its path from $z_{\text{init}} = \tau(0)$ to $z_{\text{goal}} = \tau(1)$, while some obstacles, $\mathscr{O}(t)$, potentially could interfere with its path

over the time interval T. For each vehicle, other vehicles with higher priority are considered as moving obstacles. Let the domain of τ be denoted by $S=[0,1]$; the task is then to find a path $\tau:[0,1]\rightarrow Z_{\text{free}}$. For this purpose, the VTS algorithm is proposed. The VTS algorithm is a sampling-based method that is inspired by the RRT and A* algorithms, see [1].

In the VTS algorithm, instead of randomly selecting one point in Z, several points are selected at each step. The start and goal points are $(0,0)$ and $(t,1)$, respectively. $(t,1)$ indicates the $s=1$ line, which is achieved when the vehicle has traveled the complete path τ. In order to increase the convergence rate of the method and maximize the speed of the vehicles, the random points will be generated as follows.

Definition 3 (K sample points generation): Generate $K/2$ sample points, (s,t), between lines with slopes that are functions of $v_{\min}$ and $v_{\max}$ with a small step of the nearest point in the tree, and $K/2$ on the $s=1$ line. K is a positive integer that indicates the number of sampling points.

After generating random points, $Z_{\text{new}}=(z_{\text{new},1},\ldots,z_{\text{new},K})$ will be calculated according to the vehicle dynamics. In order to apply speed limitations, if the slope of the line that connects each $Z_{\text{rand}}(i)$ and $Z_{\text{new}}(i)$ is less than the allowed value, the speed is considered to be $v_{\min}$ and if the slope is larger than the maximum allowed value, then the speed is considered to be $v_{\max}$. Note that the limitation on steer angle is applied in IRRT algorithm when steer angle is being calculated.

After finding Z_{new}, a cost function will be assigned to each of the elements of Z_{new}. The cost function H_v is defined as the distance between $Z_{\text{new}}(i)$ and z_{goal}.

The next step is to select the best $Z_{\text{new}}(i)$, z^*_{new}, the component of Z_{new} that is in Z_{free} and has the minimum cost value. The above steps are all repeated until z^*_{new} reaches the line $s=1$. The VTS algorithm is shown in Algorithm 2.

Algorithm 2 VTS Algorithm

function VELOCITY TUNING
  Tree-init($z_{\text{init}}, z_{\text{goal}}$, $K>1$, $\mathscr{G}_v=(\mathscr{V}_v,\mathscr{E}_v)$, $\mathscr{V}_v=z_{\text{init}}$, $\mathscr{E}_v=\emptyset$)
  while !flag **do**
    Generate K sample points based on Definition 3.
    for $i=1$ to K **do**
      $Z_{\text{near}}(i)=\text{Nearest}(\mathscr{G}_v=(\mathscr{V}_v,\mathscr{E}_v),Z_{rand}(i))$
      $Z_{\text{new}}(i)\leftarrow\text{Acc}(Z_{\text{near}}(i),Z_{\text{rand}}(i))$
      if !Collision-free($Z_{\text{new}}(i),Z_{\text{near}}(i)$) **then**
        $H_v(i)=\infty$
      else
        $H_v(i)\leftarrow$ distance between $Z_{\text{new}}(i)$ and $s=1$
      end if
    end for
    $n=\arg\min_i H_v(i)$
    $z^*_{\text{near}}=Z_{\text{near}}(n)$
    $z^*_{\text{new}}=Z_{\text{new}}(n)$
    $\mathscr{V}_v\leftarrow\mathscr{V}_v\cup\{z^*_{\text{new}}\}$
    if $z^*_{\text{new}}=z_{\text{goal}}$ **then**
      return flag = True
    else
      return flag = False
    end if
  end while
end function

C. Decoupled Sampling-Based Planning (DSBP) Method

By using the SRRT and VTS algorithms, the DSBP method for n vehicles is defined in Algorithm 3.

Algorithm 3 DSBP Algorithm

function MOTION PLANNING FOR MULTIPLE VEHICLE
  Init(Initial position, goal position, state limits, n)
  for $i=1$ to n **do**
    Reorder vehicle numbers based on their priority
    $\tau_i\leftarrow$ SRRT Algorithm
  end for
  for $j=1$ to n **do**
    Find configuration $\phi_i(t)$
    $\mathscr{V}_v(j)\leftarrow$ VTS Algorithm $(Z_1,\ldots,Z_{n-1})$
  end for
end function

IV. SIMULATION RESULTS

The DSBP algorithm has been tested for five vehicles in three different scenarios, "*cooperative vehicles on the highway*", "*changing lane*", and "*cooperative intersection*". In all Figs. 2–4, sub-figures (a) illustrate the explored tree during the search phase (blue curves) and the final paths computed by the SRRT algorithm (red curves). The blue stars are the start points and the red stars are the goal points. In addition, the black regions are obstacles and the white regions are obstacle-free spaces. Sub-figures (b) illustrate the $T-S$ map for one sample vehicle. The black blocks show the (t,s) points in which there will be a collision between the current vehicle and the higher-priority vehicles, and blue arcs show the tree branches produced by the VTS algorithm. In addition, the red path shows the collision free (t,s) points, which correspond to the possible fastest movement. In order to consider vehicle dimensions, the size of the vehicles has been added to the black regions. The curves in sub-figures (c) show the speed of the vehicles. Sub-figures (d) illustrate the distance between each two vehicles during the maneuvers. It has been observed that by using the DSBP algorithm, vehicles can perform the maneuvers without collision. The results show the good performance of the algorithm in the following different traffic scenarios. The parameters and values that have been used in the implementations are defined in Table I.

TABLE I: Vehicle model parameters

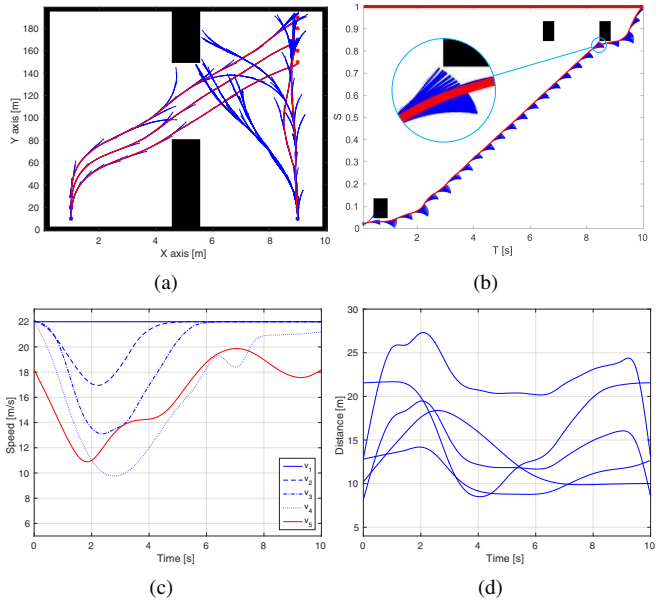
Symbol	Description	Value
L_f, L_r	Distance from the COG to the wheels	1.5 [m]
m	Mass	1700 [kg]
$C_{\alpha f}$	Cornering stiffness coefficient for front tire	13000
$C_{\alpha r}$	Cornering stiffness coefficient for rear tire	20000
δ_{max}	Maximum steering angle	$\pi/8$ [rad]
v_{min}	Minimum speed	3 [m/s]
v_{max}	Maximum speed	22 [m/s]

A. Cooperative Vehicles in Highway Maneuver

This scenario demonstrates the cooperative maneuver in highway scenario for two groups of vehicles. In this scenario, vehicles in different lanes receive a message from a construction site ahead that a lane up the road is blocked. The vehicles must communicate to form one single group before the site. This maneuver has to be done whilst prioritizing passenger safety and possible speed limits. Figure 2 demonstrates the results of using the DSBP algorithm for this scenario.

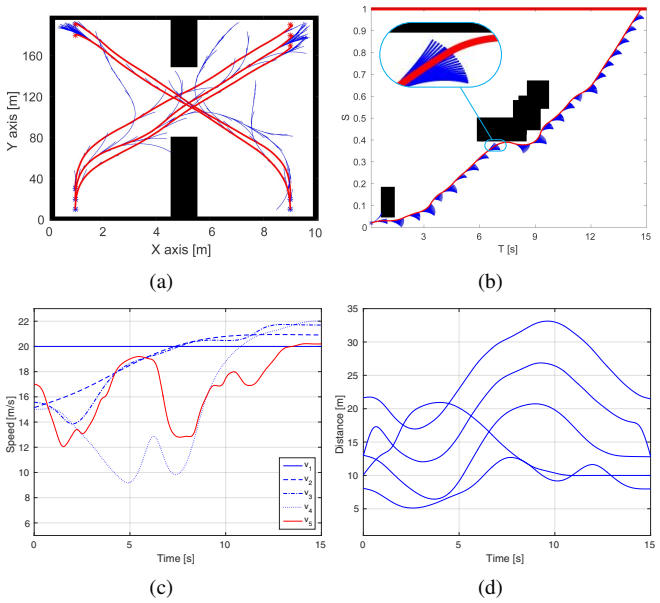
B. Changing Lane Maneuver

In this scenario, two groups of vehicles that are moving in different lanes aim to change lane. The results of applying the DSBP method for this scenario are shown in Fig. 3.



(a) (b) (c) (d)

Fig. 2: Results of the DSBP method for the cooperative vehicles in highway scenario.



(a) (b) (c) (d)

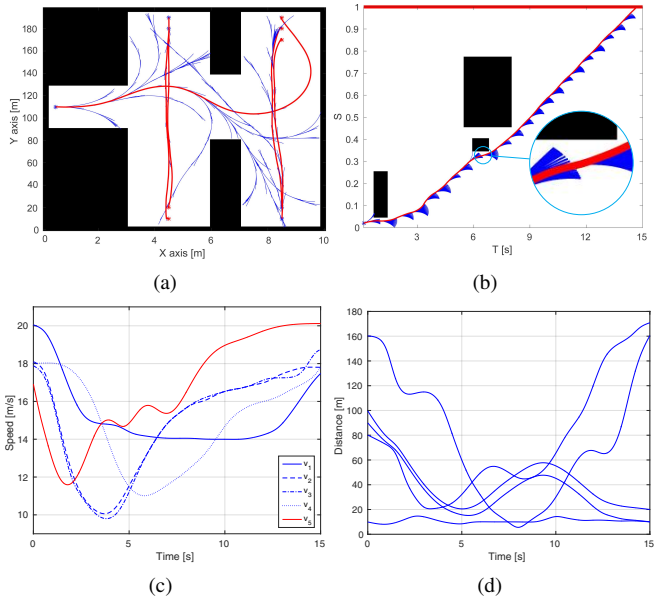
Fig. 3: Results of the DSBP method for the changing lane maneuver.

C. Cooperative Intersection Maneuver

This scenario demonstrates a complicated intersection coordination activity on a road. In this scenario, one vehicle is approaching a busy road and intends to make a left turn onto the road and collaborates with approaching vehicles to allow a comfortable and safe passage. Figure 4 shows the results.

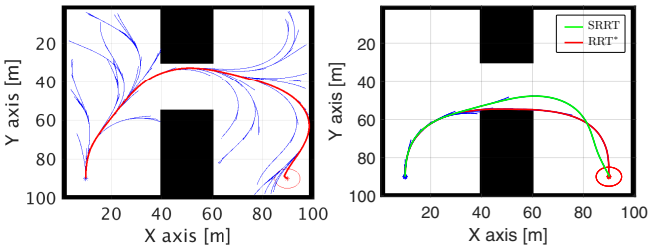
D. SRRT VS. RRT and RRT*

In order to compare the SRRT algorithm with the standard RRT and RRT* algorithms, [13], all the algorithms are used for different driving scenarios (see Figs. 5-8). Table II presents the average of the computation times of the SRRT, RRT, and RRT* algorithms for 100 runs. It can be seen from the Figs. 5-8 and Table II that RRT* provides smoother path and optimal result compared to RRT and SRRT. However, the average



(a) (b) (c) (d)

Fig. 4: Results of the DSBP method for the cooperative intersection maneuver.



(a) (b)

Fig. 5: The vehicle paths that are generated using RRT, RRT* and SRRT for U-turn maneuver.

computational time and the number of samples of RRT* is much higher than two other algorithms. Therefore, since the difference between the cost of using SRRT and RRT* is not too big, SRRT algorithm can be a good option to be used in autonomous driving. Furthermore, it was observed that RRT* works better in winding roads but, in usual urban roads and highways, where the passage is not too winding, SRRT is more efficient since the computational time is less than for the other considered methods. All implementations have been done in Matlab and on the same system, and the same biasing method has been used in all three methods. In all Figs. 5-8, sub-figures (a) illustrate the expansion of the tree and vehicle path that are generated using RRT. In sub-figures (b), the tree branches produced by the SRRT algorithm and the resulting paths that are generated using SRRT and RRT* are shown.

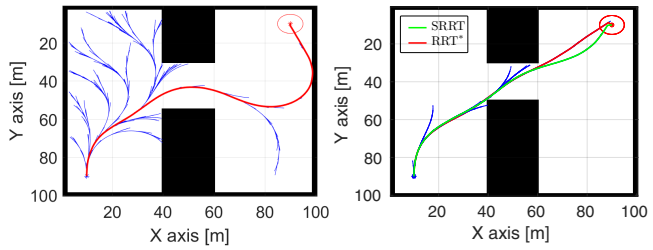
TABLE II: SRRT vs RRT and RRT*

	RRT	RRT*	SRRT
Computational time	35 [Sec]	50 [Sec]	10 [Sec]
Cost (Length)	215 [m]	114 [m]	120 [m]
Samples	650	2008	150

E. Discussion

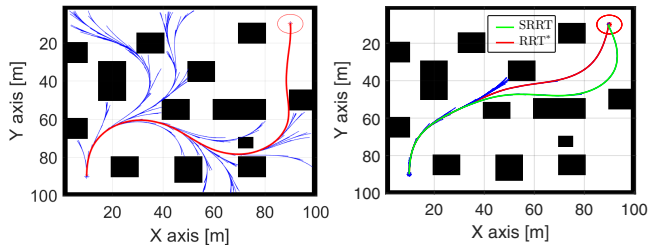
It is shown in Figs. 2–4 that by using the DSBP algorithm, the maneuvers have been done without collision.

As previously mentioned, path-planning approaches that are based on optimization have achieved great success in



(a) (b)

Fig. 6: The vehicle paths that are generated using RRT, RRT* and SRRT for changing lane scenario in an environment with a normal passage.



(a) (b)

Fig. 7: The vehicle paths that are generated using RRT, RRT* and SRRT for driving in an obstacle-rich environment.

the autonomous driving field since the methods that apply a finite set of paths can reduce the solution space; this allows for real-time implementation. In addition, optimal control-based motion-planning methods, see, e.g., [17], are more appropriate in the situations in which there are many operating constraints and actual control objectives. However, since the computational time and finding initial guess are challenges of solving an optimal-control problem, and because the sampling-based methods can handle static obstacles in an efficient way, the DSBP algorithm works better in obstacle-rich environments and for a problem that contains different operating constraints with less computational time compared to an MPC-based optimization method.

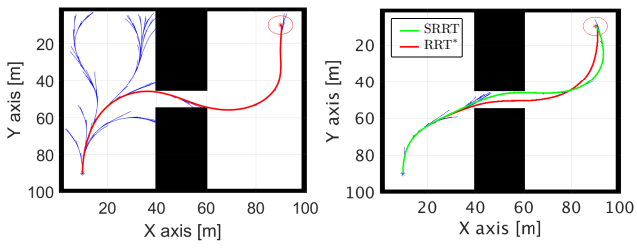
V. CONCLUSIONS

This paper presented the DSBP algorithm, a decoupled sampling-based motion-planning method, developed for multiple autonomous vehicles in order to perform different traffic maneuvers. The decoupled scheme included two steps, path-planning and motion-timing. In the path-planning part, the SRRT method was used for each vehicle, which results in a smoother path and decreases the computational time of the RRT-based methods. In the motion timing step, a semi-deterministic sampling-based method was defined that guaranteed collision avoidance between different vehicles. This motion-timing algorithm can be very useful for collision avoidance to be used with any other path-planning method. Since both the steps in the DSBP algorithm are sampling-based, it can be applied to very general dynamical models and lends itself easily to online implementation. Simulation results showed that the DSBP method works well for several different traffic maneuvers.

ACKNOWLEDGMENT

The authors would like to acknowledge Dr. Björn Olofsson for interesting discussions regarding the methods and results in this paper.

This work was partially supported by CADICS, a Linnaeus Research Environment funded by the Swedish Research Council.



(a) (b)

Fig. 8: The vehicle paths that are generated using RRT, RRT* and SRRT for changing lane scenario in an environment with a narrow passage.

REFERENCES

[1] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[2] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, June 2005.

[3] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[4] M. Kazemi, K. K. Gupta, and M. Mehrandezh, "Randomized kinodynamic planning for robust visual servoing," *IEEE Transactions on Robotics*, vol. 29, pp. 1197–1211, Oct 2013.

[5] S. M. LaValle, *From Dynamic Programming to RRTs: Algorithmic Design of Feasible Trajectories*, pp. 19–37. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[6] R. Cui, Y. Li, and W. Yan, "Mutual information-based multi-auv path planning for scalar field sampling using multidimensional RRT*," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, pp. 993–1004, July 2016.

[7] S. D. Cairano, H. E. Tseng, D. Bernardini, and A. Bemporad, "Vehicle yaw stability control by coordinated active front steering and differential braking in the tire sideslip angles domain," *IEEE Transactions on Control Systems Technology*, vol. 21, pp. 1236–1248, July 2013.

[8] X. Du, K. K. K. Htet, and K. K. Tan, "Development of a genetic-algorithm-based nonlinear model predictive control scheme on velocity and steering of autonomous vehicles," *IEEE Transactions on Industrial Electronics*, vol. 63, pp. 6970–6977, Nov 2016.

[9] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, H. E. Tseng, and F. Borrelli, "Predictive control for agile semi-autonomous ground vehicles using motion primitives," in *2012 American Control Conference (ACC)*, pp. 4239–4244, June 2012.

[10] X. Li, Z. Sun, A. Kurt, and Q. Zhu, "A sampling-based local trajectory planner for autonomous driving along a reference path," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 376–381, June 2014.

[11] S. LAVALLE, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and Computational Robotics: New Directions*, pp. 293–308, 2001.

[12] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, Aug 1996.

[13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[14] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[15] J. Wong, *Theory of Ground Vehicles*. John Wiley & Sons, 2008.

[16] E. X. Wang, J. Zou, G. Xue, Y. Liu, Y. Li, and Q. Fan, "Development of efficient nonlinear benchmark bicycle dynamics for control applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 2236–2246, Aug 2015.

[17] F. Mohseni, E. Frisk, J. Åslund, and L. Nielsen, "Distributed model predictive control for highway maneuvers," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8531–8536, 2017. 20th IFAC World Congress.