doi:10.19562/j.chinasae.qcgc.2024.09.004

基于图搜索与优化的动态非结构环境智能车辆轨迹规划*

杨秀建,白永瑞
(昆明理工大学交通工程学院,昆明 650500)

[摘要] 针对动态非结构环境下的智能车辆轨迹规划,提出了一种基于图搜索和优化的轨迹规划方法。首先,采用图搜索方法对智能车辆运动基元进行搜索,获取符合运动学特性的初始轨迹;然后,基于非线性模型预测控制方法对轨迹进行优化,以获得更平滑、更安全的轨迹。为在动态非结构环境下实现基元的快速且安全的拓展,提出了一种基元碰撞检测的方法。该方法通过障碍物膨胀和栅格离散运动基元,对非规则障碍物进行静态碰撞检测,引入速度障碍物概念,在速度空间对动态障碍物进行动态碰撞检测。在ROS/Gazebo环境下进行了算法仿真比较,并通过场地试验进行了测试评价。结果表明,相较于TEB算法,所提轨迹规划方法在满足计算实时性要求的同时,平均避障成功率提高了18%,展现出了更高的安全避障能力和可行性。

关键词:智能车辆;非结构环境;动态环境;避障

Trajectory Planning for Intelligent Vehicle in Dynamic Unstructured Environment Based on the Graph Search and Optimization Methods

Yang Xiujian & Bai Yongrui
Faculty of Transportation Engineering, Kunming University of Science and Technology, Kunming 650500

[**Abstract**] A trajectory planning method based on graph search and optimization is proposed for intelligent vehicle trajectory planning in dynamic unstructured environments. Firstly, the graph search method is employed to search for motion primitives for intelligent vehicles to obtain initial trajectories that conform to kinematic characteristics. Then, based on nonlinear model predictive control methods, the trajectory is optimized to obtain smoother and safer trajectories. In order to achieve rapid and secure expansion of primitives in dynamic unstructured environments, a method for primitive collision detection is proposed. This method uses obstacle expansion and grid discrete motion elements to perform static collision detection on irregular obstacles, and introduces in the concept of velocity obstacles to perform dynamic collision detection on dynamic obstacles in velocity space. The proposed algorithm is compared by simulations in ROS/Gazebo environment, and is evaluated by field tests. The results show that compared to the TEB algorithm, the proposed trajectory planning method improves the average obstacle avoidance success rate by 18% while meeting the real-time computing requirements, demonstrating higher safety obstacle avoidance ability and feasibility.

Keywords: intelligent vehicle; unstructured environment; dynamic environment; obstacle avoidance

*国家自然科学基金(52162046)资助。
原稿收到日期为 2024 年 03 月 16 日,修改稿收到日期为 2024 年 04 月 14 日。
通信作者:杨秀建,教授,博士,E-mail:yangxiujian@kust.edu.cn。

前言

运动规划模块是自动驾驶系统中连接感知和控制环节的关键部分,一直以来备受关注。其任务是为智能车辆规划出一条符合车辆运动约束并确保行驶安全的最优轨迹[1]。目前针对静态环境的规划问题已有较好的解决方案,但在动态环境中尤其是在参与者规则约束较弱、随机性较强的非结构化场景中,由于周围环境的复杂性和高度不确定性,轨迹规划仍然面临很多挑战[2]。

从文献报道来看,动态环境下的轨迹规划方法包括势场法、智能仿生方法、数值优化方法、基于采样以及基于图搜索的方法等。对势场法而言,通过对势场函数的改进可以适应不同的场景环境。文献[3]在规划器中引入了考虑障碍物和道路边界的势力场,实现了车辆的有效避障。文献[4]通过在势场函数中添加位置和速度虚拟力,实现了智能网联汽车在道路环境下的无振荡运动轨迹与速度规划。势场法的主要问题是易陷入局部最小,且不能对智能车辆的非完整性约束进行表达。一些学者研究了基于智能仿生方法的动态环境下的避障问题。在这方面,文献[5]在粒子群算法中引入了惯性率的概念,显著增强了算法性能,提高了避障能力。文献[6]中通过改进避障策略,提出了一种双优化蚁群算法,以提升收敛速度和避障能力。文献[7]中提出了一种基于时间窗约束的改进遗传算法,增强了算法的稳定性和收敛速度。总体而言,智能仿生方法在实时性和稳定性方面还存在一定的不足,难以完全适应复杂的动态环境。在基于数值优化方法方面,文献[8]中提出了动态障碍函数(dynamic control barrier function)的概念,并将其作为安全距离约束加入到模型预测控制中,通过非线性求解器求得的最优避障轨迹能够与障碍物始终保持一定的安全距离。文献[9]通过在模型预测控制器中引入车辆避障惩罚函数,实现了智能车的换道避障轨迹规划。在考虑安全性的模型预测控制框架下,文献[10]中提出了一种近似凸优化方法,提升了整个优化过程的效率。基于优化的方法直观准确、适应性强,但须对初始值进行适当的设置。基于采样的方法有快速探索随机树(RRT)和动态窗口法(DWA)等方法,这些算法通过采样点评估轨迹的可行性,并寻找可行的轨迹。文献[11]中在RRT算法的基础上引入了时间信息,提出了基于时间的快速探索随机树(TB-RRT),以满足在动态环境中在指定时间内到达目标点的任务需求。尽管基于采样的方法具有概率完备性且能够有效处理高维规划问题,但是由于其随机采样的特性,找到的轨迹通常不是最优的,且在动态环境下可能会面临收敛过慢等问题。典型的基于图搜索的方法包括A*和混合A*算法,原本基于图搜索的方法只能处理静态环境下的规划问题,但通过构建时空栅格地图以考虑障碍物未来的运动状态,能够扩展图搜索方法用于动态环境的轨迹规划[12-14]。尽管图搜索算法生成的轨迹接近全局最优解,但由于状态空间的离散化,轨迹可能不连续,且时空栅格地图的构建耗时较长,会导致规划效率降低。

由上述研究分析看出,动态环境下的轨迹规划还面临着很多挑战。首先,规划出的轨迹必须遵循智能车辆自身的运动学和动力学限制,否则无法有效执行;其次,在动态环境中,感知和控制结果的不确定性要求规划算法能够高频且稳定地生成轨迹;最后,在动态环境中,未考虑时间维度的算法表现较差,而直接考虑时间维度并建立时空栅格的算法会导致计算速度下降,从而降低规划的实时性。

针对以上问题,本文提出了一种基于图搜索与优化于一体的轨迹规划方法。首先,对机器人的控制空间进行离散化,并通过向前模拟得到扩展的运动基元树;在拓展运动基元树的过程中,针对动态非结构化环境,采用改进的碰撞检测方式对运动基元进行碰撞检测,以确保运动基元树的快速安全拓展;然后,基于A*算法对运动基元树进行搜索,获取初始轨迹;最后,在初始轨迹的基础上对轨迹进行优化。本文所提方法充分考虑智能车辆的运动学限制,通过引入过程运动规划(partial motion planning,PMP)[15]机制,保证整个搜索算法的实时性,并避免了机器人在动态环境中可能发生的死锁现象,可以在不直接考虑时间维度的情况下高频稳定地输出轨迹。

1　算法结构

本文提出一种分层轨迹规划方法,目的是在动态非结构环境下高频稳定地生成更加安全和平滑的行驶轨迹,具体原理如图1所示。

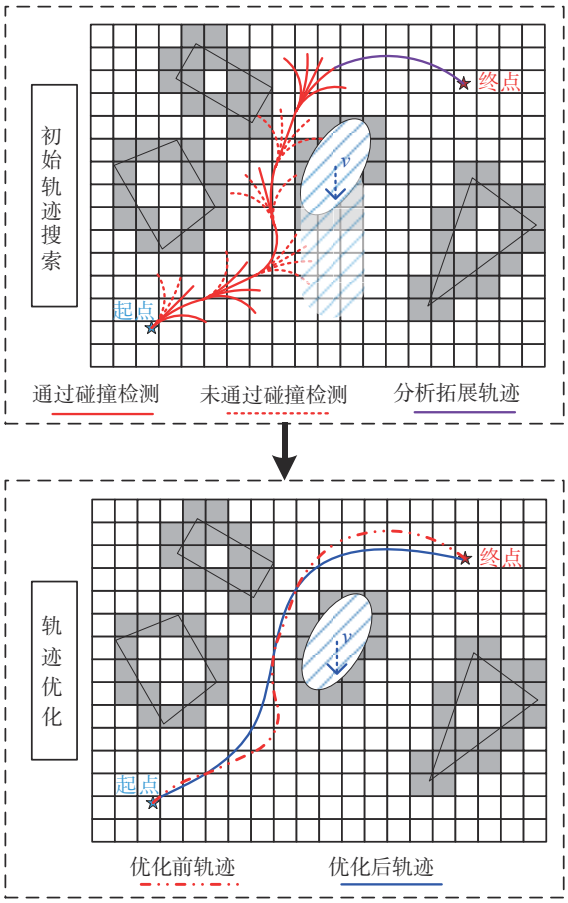


图 1 算法结构示意图

算法总体包括初始轨迹搜索和轨迹优化两个过程。其中,初始轨迹搜索过程采用图搜索方法,通过不断构建运动基元树完成节点的拓展,最终在状态栅格内找到一条时间和控制成本最小的轨迹。在搜索过程中,由于时空栅格地图构建耗时较长且需要对动态障碍物进行长时间的精准预测,本文采用二维栅格地图对环境进行描述。为适应动态非结构环境,对节点的拓展方式进行改进,并提出了一种新的碰撞检测方式,以确保节点在复杂动态非结构环境下能够安全快速拓展。由于初始轨迹是在状态空间和控制空间离散化的基础上生成的,因此往往表现出较为明显的非平滑性,难以满足跟踪控制环节的执行需求。此外,在初始轨迹搜索阶段,未考虑与障碍物之间的距离信息,导致初始轨迹通常接近障碍物,存在一定的安全隐患。为提高轨迹的平滑性和安全性,轨迹优化过程采用数值优化方法对初始轨迹进行进一步优化。由于提供了初始轨迹值作为初始条件,整个数值优化过程能够迅速收敛至最优解,从而快速得到车辆最终的行驶轨迹,驱动车辆不断向目标点驶去。

2 初始轨迹搜索

2.1 车辆运动学建模与车辆运动基元

基于如图 2 所示的自行车模型对智能车辆的运动学进行建模,具体表达为

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \dfrac{v\tan\delta}{L} \\ \dot{v} = a \end{cases} \tag{1}$$

式中:x、y 和 θ 分别代表世界坐标系下车辆后轴中心的 x 坐标、y 坐标和方向角;v 表示车辆的纵向速度;L 为车辆轴距;δ 和 a 分别表示车辆的横向控制量(前轮转角)和纵向控制量(加速度),其中前轮转角的正负按右手法则确定。

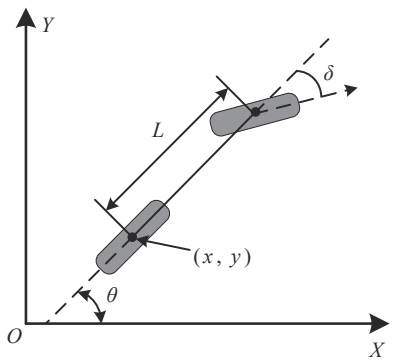


图 2 车辆运动学模型

对控制量 δ 和 a 分别在区间 $[-\delta_{\max}, \delta_{\max}]$ 和 $[-a_{\max}, a_{\max}]$ 内进行离散化,得到离散化值 $\delta_N = \{\delta_1, \cdots, \delta_N\}$ 和 $a_N = \{a_1, \cdots, a_N\}$,并向前模拟长度为 τ 的时间,获得智能车辆的运动基元,即在给定运动时间 τ 内的一段较短的轨迹。通过在状态空间中持续扩展运动基元,能够在遵循车辆运动约束的条件下,对状态空间进行充分的探索。图 3 展示了离散化前轮转角 $\delta = \{-\delta_{\max}, -\delta, 0, \delta, \delta_{\max}\}$ 和加速度 $a = \{-a_{\max}, 0, a_{\max}\}$ 的单层运动基元。其中,每个圆弧段上的圆点表示基于对应的转角和加速度采样生成的运动基元的最终状态。

定义车辆的状态空间 $\boldsymbol{s} = [x, y, \theta, v, t]^{\mathrm{T}}$,给定时刻 k 的状态 $\boldsymbol{s}_k = [x_k, y_k, \theta_k, v_k, t_k]^{\mathrm{T}}$ 和对应的离散控制量 δ 和 a,则下一时刻的状态 $\boldsymbol{s}_{k+1}$ 表示为

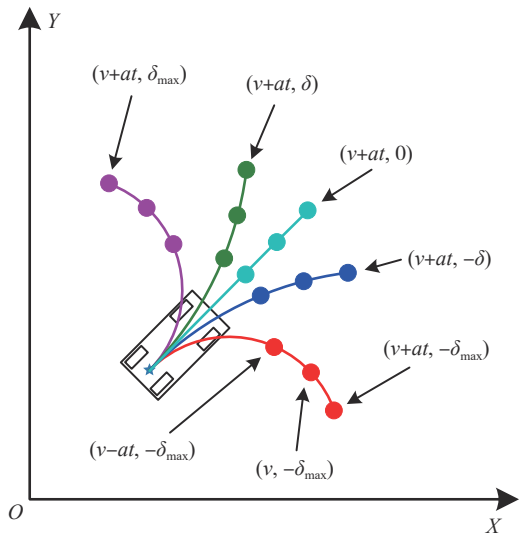


图3　单层运动基元示意图

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \\ t_{k+1} \end{bmatrix} = \begin{cases} \begin{bmatrix} x_k + d\cos\theta_k \\ y_k + d\sin\theta_k \\ \theta_k \\ v_k + a\tau \\ t_k + \tau \end{bmatrix}, & \delta = 0 \\ \begin{bmatrix} x_k + \dfrac{L(\sin\theta_{k+1} - \sin\theta_k)}{\tan\delta} \\ y_k - \dfrac{L(\cos\theta_{k+1} - \cos\theta_k)}{\tan\delta} \\ \theta_k + \dfrac{d\tan\delta}{L} \\ v_k + a\tau \\ t_k + \tau \end{bmatrix}, & \delta \neq 0 \end{cases} \tag{2}$$

式中:d为拓展时间内车辆的位移;τ为起始状态和终点状态之间的拓展时间步长。

在相邻状态之间进行拓展时,按照固定时间间隔Δt和最小拓展步长$d_{\min}$进行拓展,即

$$\tau = \begin{cases} \dfrac{\sqrt{v_s^2 + 2ad_{\min}} - v_s}{a}, & 0 < |d| \leqslant d_{\min} \\ \Delta t, & |d| > d_{\min} \end{cases} \tag{3}$$

由于本文使用网格对运动基元进行剪枝,为尽可能避免拓展的相邻节点落到一个网格内,$d_{\min}$取单个网格的对角线长度。

此外,在相邻节点拓展时,线速度v可能会在拓展时间步长τ耗尽前就达到了最大值。在这种情况下,运动基元的剩下部分以最大速度生成,且此时须重新计算车辆的行驶位移d。运动基元的完整生成过程可通过算法1中的伪代码进行描述,其中只有通过碰撞检测的运动基元才被视为有效,进而成为搜索过程中的一个节点。

算法 1: 运动基元生成

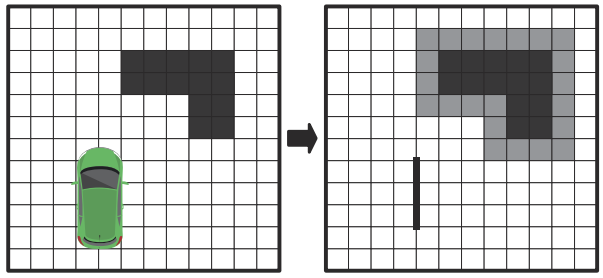
输入: 开始状态s_k
输出: 有效的运动基元集合 *primitives*

```
primitives ← ∅;
for δ ∈ δ_N do
    for a ∈ a_N do
        τ ← CalculateTimeStep();
        // 由式(2)计算下一时刻的状态
        s_{k+1} ← MotionPrimitives(s_k, τ, δ, a);
        // 进行碰撞检测
        if CollisionFree(s_k, s_{k+1}, δ, a) then
            s_{k+1}.g ← s_k.g + PrimitiveCost();
            s_{k+1}.h ← Heuristic(s_{k+1});
            s_{k+1}.f ← s_{k+1}.g + s_{k+1}.h;
            s_{k+1}.t ← s_k.t + τ;
            s_{k+1}.parent ← s_k;
            Insert(s_{k+1}, primitives);
return(primitives);
```

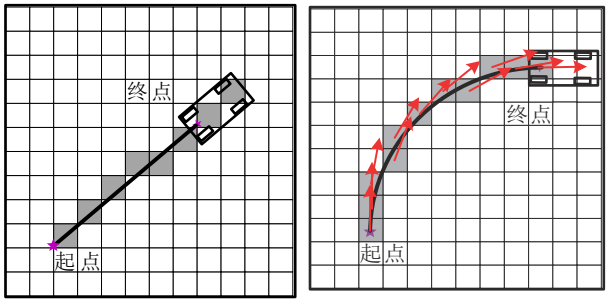
2.2　碰撞检测

碰撞检测主要包括两步。首先利用栅格地图进行静态障碍物的碰撞检测,如图4(a)所示,在栅格地图构建时,将栅格地图以车宽的一半进行膨胀,从而可以用一条线段在栅格地图上对车辆进行有效表示。而在栅格上对线段的碰撞检测已有较为成熟的方法[16],从而可以迅速地对任意时刻的车辆状态进行检查。对于拓展出来的运动基元,同样可以利用栅格进行离散,并进行碰撞检测。如图4(b)所示,当$\delta = 0$即车辆做直线运动时,可以根据式(2)计算出运动基元的起点和终点,再在终点位置加上车身长度,然后再次利用文献[16]中介绍的方法进行直线上的碰撞检测。当$\delta \neq 0$时,此时可以认为车辆在做半径和转角已知的圆弧运动,此时可借助计算机图形学上的绘制圆弧算法对轨迹进行栅格上的离散并检查占据状态,具体如图4(c)所示,其中每个栅格上的红色箭头表示对应栅格上的车身朝向。

在进行静态碰撞检测时,除对拓展出的运动基元进行碰撞检测外,为确保最终搜索到的轨迹更加安全,还希望每一个时刻的状态都不属于不可避免碰撞状态(inevitable collision states, ICS)[17]。ICS被定义为一种状态,一旦智能车辆进入该状态,就必然在一段时间内与障碍物发生碰撞。然而,ICS的识别是一个耗时的过程,因此本文对该过程进行了简化,即在每个运动基元的终点状态上,以最大减速度进行一段时间的额外基元拓展,并对这段额外拓展的基元进行碰撞检测,如果在这个额外拓展中发生了碰撞,也认为静态碰撞检测失败。



(a) 栅格膨胀过程



(b) 直线运动情况 (c) 圆弧运动情况

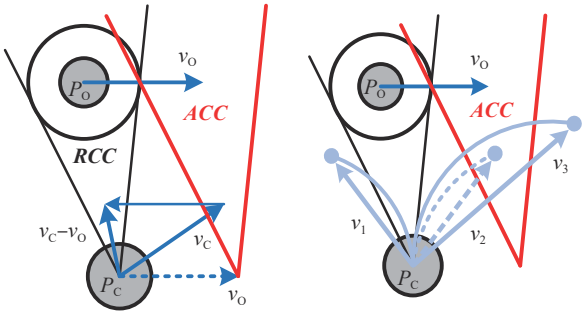
图4 静态碰撞检测示意图

当静态碰撞检测通过后,才进行第2步的动态碰撞检测。本文借助速度障碍物方法中的速度障碍物来对动态障碍物进行碰撞检测。速度障碍物是指速度空间中的障碍物,如图5(a)所示的速度空间中,P_O、P_C分别代表移动障碍物和智能车辆。假设P_O的运动速度为零($v_O = 0$),当智能车辆下一时刻的运动速度区间落入相对碰撞区(relative collision cone, RCC)时,此时智能车径直驶向障碍物,从而存在碰撞的可能性。而当v_O不为零时,若再次希望确定智能车可能发生碰撞的速度区间,可以通过以下数学表达式来确定:

$$\boldsymbol{ACC} = \boldsymbol{RCC} \oplus v_O \tag{4}$$

式中:$\boldsymbol{ACC}$(absolute collision cone)称为绝对碰撞区,表示智能车辆与移动障碍物之间可能发生碰撞的速度v_C的集合,即速度障碍物;$\oplus$表示闵可夫斯基矢量和运算。

通过速度障碍物,可以对未来一段时间内的危险速度区间进行判断。在进行动态碰撞检测时,如果某个拓展的基元速度落入$\boldsymbol{ACC}$区域,就判定该基元的动态碰撞检测失败,也意味着该基元的碰撞检测失败,因此该基元不予拓展。如图5(b)所示,v_1和v_3位于$\boldsymbol{ACC}$区域外,表示这两个运动基元通过了动态碰撞检测,这意味着智能车辆可以选择从动态障碍物行进的后方绕行通过障碍物,或通过加速来快速远离障碍物;而v_2落在$\boldsymbol{ACC}$区域内,该基元的碰撞检测不通过,因此不会将该基元作为搜索的节点之一。由于速度障碍物是对未来时刻的一种理想的表示形式,即运动障碍物不一定会一直以当前观测到的速度做匀速运动,所以只对时间阈值t_{thr}内的运动基元进行动态碰撞检测,即拓展的基元的状态空间$\boldsymbol{s} = [x, y, \theta, v, t]^T$中的$t < t_{thr}$时才进行动态碰撞检测。



(a) 速度障碍物 (b) 动态碰撞检测

图5 动态障碍物检测示意图

2.3 代价函数和启发函数

在初始轨迹搜索阶段,此时的目标是寻找一条时间和能量最优的轨迹,整条轨迹的代价定义如下:

$$J(T) = \int_0^T \sqrt{(a^2 + \delta^2)}\,\mathrm{d}t + \rho T \tag{5}$$

式中:T表示整条轨迹的总时间;ρ是时间T的权重系数。在这一定义下,每个离散控制量得到的运动基元的代价函数可表示为$\left(\sqrt{a^2 + \delta^2} + \rho\right)\tau$。因此,从起始状态到某个拓展了$N$个基元的节点的总代价值为$\sum_{n=1}^{N}\left(\sqrt{a_n^2 + \delta_n^2} + \rho\right)\tau_n$。

对于搜索算法,合适的启发函数可以引导拓展节点快速向目标点拓展,从而加快搜索过程。对于智能车辆而言,选择从当前节点到目标位置的Reeds-Sheep曲线长度作为启发函数是合适的,因为该曲线考虑了车辆的运动学约束,更加符合实际行驶情况,但是该曲线是不考虑障碍物的最短路径曲线,在非结构环境中可能存在横亘在智能车辆前方的长杆形障碍物。在这种情况下,如果不考虑障碍物并将Reeds-Sheep曲线长度作为启发函数,就不太合理了。因此,为考虑非结构环境下的障碍物,本文使用Reeds-Sheep曲线长度和Dijkstra距离的最大值作为启发函数值。

2.4　初始轨迹搜索算法流程

以车辆当前状态$(x_0, y_0, \theta_0, v_0, t_0)$为初始状态,进行车辆运动基元的拓展,同时在拓展过程中对获得的运动基元进行碰撞检测,以确保运动基元的安全性。考虑到动态环境下规划算法需要高频给出规划结果,本文引入了PMP机制,即不把搜索到达终点作为搜索截止的唯一条件,还考虑了搜索耗时。当搜索耗时达到截止要求时,算法会返回当前搜索到的无碰撞轨迹。整个搜索流程可以用算法2中的伪代码进行描述,其中OPEN是一个优先队列,类似于A*算法中的开集,通过该优先队列对已探索节点进行排序和弹出。HMAP是一个哈希表,以键值对的形式实现对状态栅格中存储节点的快速查找。为迅速高效地探索配置空间,所以需要对拓展节点进行剪枝操作,即一个状态栅格内只保留最优的一个节点,具体的剪枝方式参照文献[18]中的方法。上述流程不断进行,直至任何一个节点到达终点或分析拓展(analytic expansion)成功或到达了最大搜索时间。

算法 2: 初始轨迹搜索

```
输入: 初始状态s_0,目标位置x_g
输出: 初始轨迹点集合 P
Initialize();
OPEN.add(s_0), HMAP.add(s_0);
while OPEN is not empty do
    s_k ← OPEN.pop();
    s_k.closed ← true;
    if ReachGoal(s_k) ∨ AnalyticExpand(s_k) then
        return BackTrack(s_k);
    if ReachTime() then
        return BackTrack(s_k);
    // 拓展车辆运动基元
    primitives ← GeneratePrimitives(s_k);
    for s_{k+1} in primitives do
        key ← GetKey(s_{k+1});
        // 检查状态栅格是否被探索过
        if HMAP.find(key) ≠ ∅ then
            Prune(s_{k+1});      // 被探索过, 执行剪枝
        else
            OPEN.add(s_{k+1}), HMAP.add(s_{k+1});
```

2.5　分析拓展和重规划策略

在节点拓展时,由于控制输入的离散化,很难使原始轨迹末端精确地到达目标状态。为了弥补这一缺陷并加速搜索速度,引入了分析扩展策略。具体而言,在从优先队列中弹出一个节点时,采用抽样的方式基于Reeds-Sheep曲线将弹出节点的姿态(x, y, θ)和目标姿态(x, y, θ)进行连接。如果该Reeds-Sheep曲线通过了静态碰撞检测,搜索过程将提前停止,并为Reeds-Sheep曲线中的路径点分配速度信息。路径点的速度分配过程如下:首先,根据Reeds-Sheep曲线中的路径点数量以及当前节点状态中的速度信息和终点的速度信息计算相邻路径点之间的速度差值;然后,以匀减速的方式,按照速度差值为间隔依次为每个路径点分配速度信息。如图1中初始轨迹搜索过程中末端的紫色线段即通过分析拓展得到的轨迹线段。

其次,为更好地适应不断变化的动态环境,整个规划算法在固定时间间隔调用的同时,若当前规划出的轨迹与移动障碍物发生碰撞,也将立即进行轨迹的重规划。这确保一旦检测到碰撞,系统能够迅速应对,并采用新的安全轨迹。

3　轨迹优化

由于初始轨迹是通过搜索获得的,因此所得轨迹往往不是最优的,且可能存在较大的非平滑性,难以满足跟踪控制环节的执行需求。同时,在初始轨迹搜索过程中未考虑障碍物的距离信息,导致所搜索到的轨迹通常较为接近障碍物,从而导致在动态环境中规划的安全性相对较低。因此,本文设计并采用了考虑障碍物距离的非线性模型预测控制(nonlinear model predictive control, NMPC)方法对初始轨迹进行优化,以使初始轨迹在光滑性和安全性方面得到改善。

在NMPC中,仍然采用图2所示的自行车模型进行运动学描述。选取状态向量$\boldsymbol{x}=[x, y, \theta, v]^{\mathrm{T}}$,控制输入$\boldsymbol{u}=[a, \delta]^{\mathrm{T}}$,则系统的离散状态方程为

$$\boldsymbol{x}_{k+1}=f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{6}$$

式中:$\boldsymbol{x}_k$、$\boldsymbol{x}_{k+1}$分别为系统在k和k+1时刻的状态向量;f是描述系统状态演化的非线性函数。

3.1　优化目标函数

对初始轨迹进行优化,期望使其更加平滑并在一定程度上远离障碍物,以增强轨迹的安全性。为此,定义优化目标函数为

$$J=(\boldsymbol{x}_N-\boldsymbol{x}_{\mathrm{ref},N})^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{x}_N-\boldsymbol{x}_{\mathrm{ref},N})+\sum_{i=0}^{N-1}(J_{x,i}+J_{u,i})+\lambda J_{\mathrm{obs}} \tag{7}$$

式中:右边第1项为最后一个轨迹点的终端惩罚项,矩阵$\boldsymbol{P}$为对应的终端权重矩阵;$\boldsymbol{x}_{\mathrm{ref},N}$为最后一个轨迹点的参考初始轨迹点;$J_{x,i}$为第$i$个预测时刻的状态惩罚项,旨在减小与初始轨迹的偏差;$J_{u,i}$为第i个

预测时刻的控制惩罚项,目的是使初始轨迹平滑;最后一项为碰撞惩罚项,以确保轨迹点远离障碍物,从而提升整体轨迹的安全性;λ为碰撞惩罚项的权重系数。其中,$J_{x,i}$定义为

$$J_{x,i}=(\boldsymbol{x}_i-\boldsymbol{x}_{\mathrm{ref},i})^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}_i-\boldsymbol{x}_{\mathrm{ref},i}) \tag{8}$$

式中:$\boldsymbol{x}_i$为第i个预测时刻的状态变量;$\boldsymbol{x}_{\mathrm{ref},i}$为第$i$个预测时刻参考的初始轨迹点;$\boldsymbol{Q}$为状态权重矩阵。控制惩罚项可以表示为

$$J_{u,i}=\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}_i \tag{9}$$

式中:$\boldsymbol{u}_i$为第i个预测时刻的控制输入;$\boldsymbol{R}$为控制输入权重矩阵。

碰撞惩罚项定义为障碍物施加在每个轨迹点上的斥力,可以用下式表示:

$$J_{\mathrm{obs}}=\sum_{k=1}^{N}F\left(d(\boldsymbol{x}_k)\right) \tag{10}$$

式中:$\boldsymbol{x}_k$表示第k个预测时刻的状态变量,也就是第k个轨迹点;$d(\boldsymbol{x}_k)$表示第k个轨迹点和最近障碍物的距离;F为斥力场函数,定义如式(11)所示。

$$F\left(d(\boldsymbol{x}_k)\right)=\begin{cases}\left(d(\boldsymbol{x}_k)-d_{\mathrm{thr}}\right)^2, & d(\boldsymbol{x}_k)\leqslant d_{\mathrm{thr}}\\ 0, & d(\boldsymbol{x}_k)>d_{\mathrm{thr}}\end{cases} \tag{11}$$

式中d_{thr}为距离惩罚阈值,即只有在轨迹点与最近障碍物的距离小于惩罚阈值时才施加斥力。

3.2 约束条件

为确保经过优化后的轨迹是可行的,需要对整个优化过程设置约束条件,具体如下。

(1)系统动力学约束:保证每个轨迹点都符合系统动力学模型,即

$$\boldsymbol{x}_{k+1}=f(\boldsymbol{x}_k,\boldsymbol{u}_k),\quad k=0,\cdots,N-1 \tag{12}$$

式中$\boldsymbol{x}_k$和$\boldsymbol{x}_{k+1}$分别代表第k和第k+1时刻的状态变量。

(2)起始点约束:

$$\boldsymbol{x}_0=[\,x_{\mathrm{s}}\quad y_{\mathrm{s}}\quad \theta_{\mathrm{s}}\quad v_{\mathrm{s}}\,]^{\mathrm{T}} \tag{13}$$

式中:$\boldsymbol{x}_0$代表第一个状态向量,也就是初始点;x_{s}、y_{s}、θ_{s}和v_{s}分别为搜索起始状态点对应的x坐标、y坐标、方向角和纵向速度。

(3)控制输入范围约束:确保纵向加速度和前轮转角都在限制范围内,即

$$\boldsymbol{u}_{\min}\leqslant\boldsymbol{u}_k\leqslant\boldsymbol{u}_{\max},\quad k=0,\cdots,N-1 \tag{14}$$

式中$\boldsymbol{u}_k$代表第k个预测时刻的控制输入。

(4)纵向速度范围约束:

$$v_{\min}\leqslant v_k\leqslant v_{\max},\quad k=1,\cdots,N \tag{15}$$

式中v_k为第k个预测时刻时的纵向速度。

(5)终点约束:

$$\begin{bmatrix}x_{\mathrm{g}}-\varepsilon\\ y_{\mathrm{g}}-\varepsilon\\ \theta_{\mathrm{g}}-\varepsilon\\ v_{\mathrm{g}}-\varepsilon\end{bmatrix}\leqslant\boldsymbol{x}_N\leqslant\begin{bmatrix}x_{\mathrm{g}}+\varepsilon\\ y_{\mathrm{g}}+\varepsilon\\ \theta_{\mathrm{g}}+\varepsilon\\ v_{\mathrm{g}}+\varepsilon\end{bmatrix} \tag{16}$$

式中:x_{g}、y_{g}、θ_{g}和v_{g}分别为搜索终点对应的x坐标、y坐标、方向角和纵向速度值;ε为松弛因子,是一个很小的正数,通过引入松弛因子,使得优化问题可以在一定程度上轻微违背终端约束,进而加快优化求解过程。

4 仿真与试验评价

4.1 仿真分析

在机器人操作系统(robot operating system,ROS)框架下,通过C++编程语言对算法进行实现,并基于Gazebo仿真平台搭建了仿真环境,对本文提出的轨迹规划算法进行仿真评价。仿真中智能车辆的定位信息和动态障碍物的运动信息均通过直接读取仿真环境的信息获得。轨迹规划算法所需的二维栅格地图是通过开源的costmap_2d功能包、对仿真环境中的16线激光雷达数据进行处理实时构建得到。仿真试验相关参数如表1所示。

表1 仿真试验相关参数

参数	数值
车长/m	1.92
轴距L/m	1.33
车宽/m	1.02
单个栅格边长/m	0.20
剪枝网格边长/m	0.20
拓展步长τ/s	0.8
速度范围/(m·s^{-1})	[-3, 3]
最大搜索时间/ms	100
离散加速度集合/(m·s^{-2})	{-0.5, -0.25, 0.0, 0.25, 0.5}
离散前轮转角集合/(°)	{-28, -21, -14, -7, 0, 7, 14, 21, 28}

仿真中将本文算法与在非结构环境中常用的时间弹性带(time-elastic bands,TEB)算法进行比较,两种算法都充分考虑了车辆的运动学约束,且都对动态障碍物进行了预测和处理。仿真基于如图6所示的动态非结构环境,长、宽均为10 m,环境中的立方体和圆柱体为静态障碍物,行人为动态障碍物,以1.5 m/s的速度匀速运动。行人A_1在(4 m, 1 m)和(4 m, -5 m)两点间做往复直线运动,行人A_2在(7 m,

5 m)和(7 m, −2 m)两点间做往复直线运动。轨迹规划的起点为(0 m,0 m),终点为(9 m,1 m)。TEB规划算法所用的车辆参数、速度范围和加速度范围均与表1中相应的参数一致。

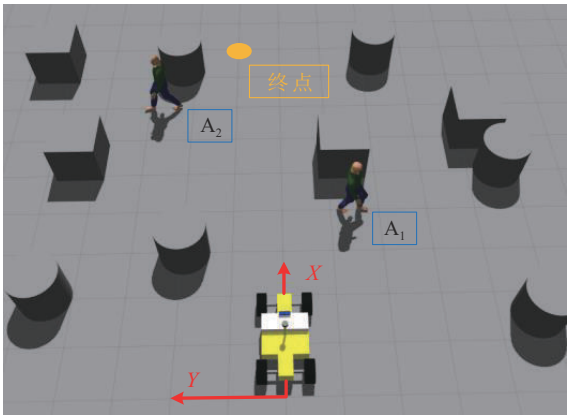


图6　仿真动态环境

图7给出了在该环境下本文算法和TEB算法的行驶轨迹。可以看出,本文算法车辆直接朝向目标点运动,而TEB算法车辆在面对动态障碍物时经历了一个反应式避障的过程,因此导致整条轨迹存在一定的弯折。

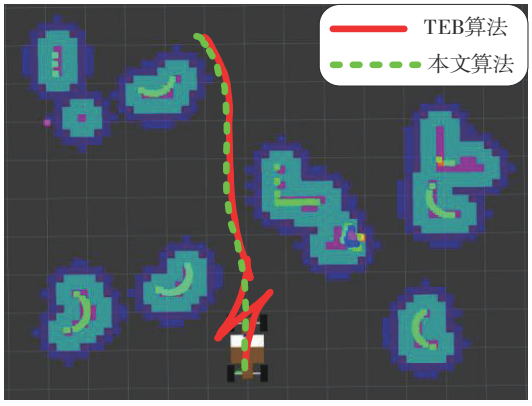
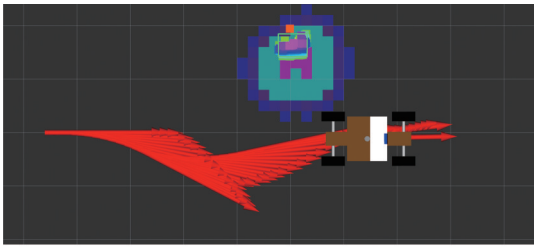
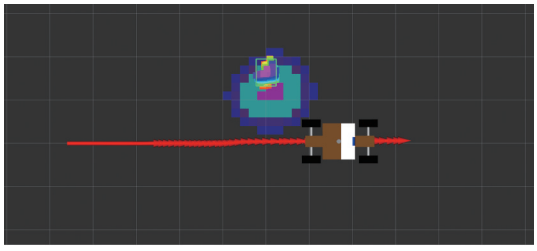


图7　行驶轨迹对比图

为更详细展示本文算法对动态障碍物的避障策略,选择单障碍物场景做进一步说明。目标点为智能车辆的正前方,环境中有一个从车辆左侧向右侧移动的动态障碍物,具体场景和轨迹规划结果如图8所示。图8(a)和图8(b)分别给出了TEB算法和本文算法规划得到的轨迹。图中的每个红色箭头代表车辆在特定时刻的方位角信息,通过这种方式能够直观地展示整个轨迹的变化过程。可以看出,TEB算法在面对前方障碍物时往往表现为一定程度的反应式避障,而本文算法则总是能够在障碍物接近时灵活选择停车等待或加速通过,使整条轨迹呈直线行驶状态。这是因为本文算法在节点拓展时,通过动态碰撞检测过程,可以对潜在不安全的节点进行剔除,从而可以实现一定程度上将决策融入到规划中。相较之下,TEB算法虽然同样考虑了动态障碍物,但其将动态障碍物视作匀速运动,以预测动态障碍物的未来轨迹。TEB算法不仅规避当前时刻的障碍物,还对动态障碍物未来的轨迹进行规避,导致整体轨迹出现向障碍物移动方向的偏转。



(a) TEB算法行驶轨迹

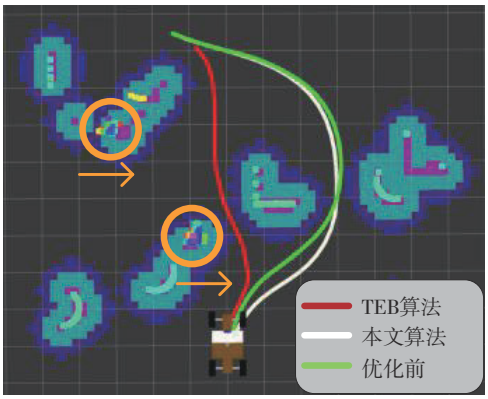


(b) 本文算法行驶轨迹

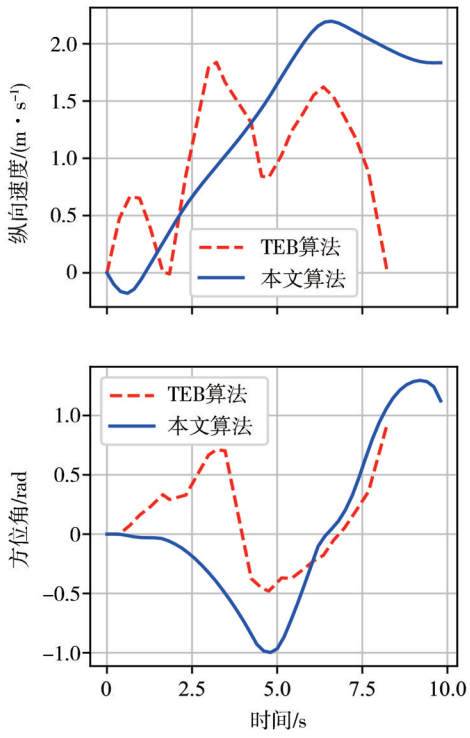
图8　单障碍物场景下行驶轨迹对比

图9给出了某一时刻本文算法和TEB算法规划出来的轨迹比较。图9(a)中白色曲线为本文算法规划的轨迹,绿色曲线为本文规划算法中未经NMPC优化的轨迹,红色曲线表示TEB算法规划的轨迹,橙色圆圈中的障碍物为移动障碍物,箭头指向其运动方向。图9(b)给出了轨迹的详细信息,包括纵向速度和方向角的规划。可以看出,由于动态障碍物向右方运动阻挡了智能车辆的前进方向,本文规划算法会选择加速绕行以远离动态障碍物来避免发生碰撞。比较优化前和优化后的轨迹,可以发现优化后轨迹更加平滑和远离障碍物,保证了整条轨迹执行时更加安全。相比之下,作为一种时间最优的算法,TEB算法在任何情况下总是选择穿过靠近动态障碍物的区域,导致其更容易与动态障碍物发生碰撞。

为进一步评价动态障碍物数量对算法的影响,



(a) 不同轨迹展示



(b) 纵向速度和方位角规划

图 9 某一时刻规划的轨迹比较

设定了如图 10 所示的 8 m × 10 m 的动态仿真环境，仍然将仿真行人作为移动障碍物。每个仿真行人的初始位置和运动终点位置在每次进行新的规划时在仿真环境中随机生成，但设定每个仿真行人的初始位置和运动终点位置之间的距离大于仿真环境(8 m × 10 m)对角线的一半，以确保每个动态行人都能对智能车辆产生一定的阻碍。规划起点和终点分别为(0 m，0 m)和(9 m，0 m)，在规划过程中仿真行人以 1 m/s 的速度在初始位置和运动终点位置之间做往复直线运动。规定在规划过程中，车辆不主动与动态障碍物发生碰撞，且行驶耗时小于 60 s 才算规划成功。反之，如果车辆主动与障碍物发生了碰撞或行驶超时，都认为规划失败。在试验过程中，不断增加动态障碍物的数量，每一种数量下各算法都进行 30 次试验。

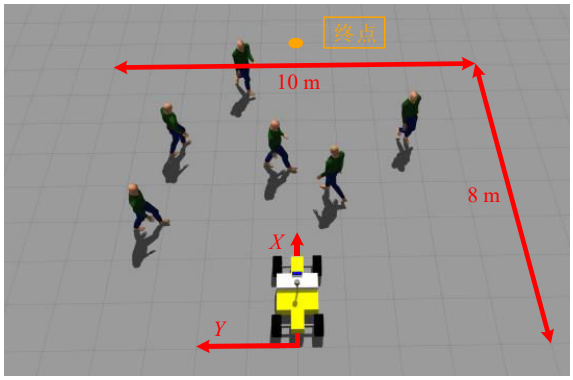


图 10 动态障碍物数量影响的仿真场景

图 11 展示动态障碍物数量对本文算法和 TEB 算法避障成功率的影响比较。容易看出，动态障碍物数量对 TEB 算法避障成功率的影响更大，当障碍物数量增加到 4 个时，TEB 算法的避障成功率迅速下降至 57%，而本文算法的避障成功率仍能保持在 83%；当动态障碍物数量从 4 个增加至 6 个，TEB 算法的避障成功率略有下降，本文算法的避障成功率出现了一定幅度的下降，但仍显著高于 TEB 算法。这是因为此时环境变得更为混乱和复杂，导致本文算法倾向于选择停车等待，智能车辆表现出一定的谨慎，因而对规定时间内到达终点产生一定的影响。而 TEB 算法作为一种追求时间最优的算法，虽然其路径选择常常靠近障碍物，降低了轨迹的安全性，但对于在规定的时间内到达目标点是有利的。总体而言，本文算法在各种动态障碍物数量的情况下，避障成功率都要显著高于 TEB 算法。

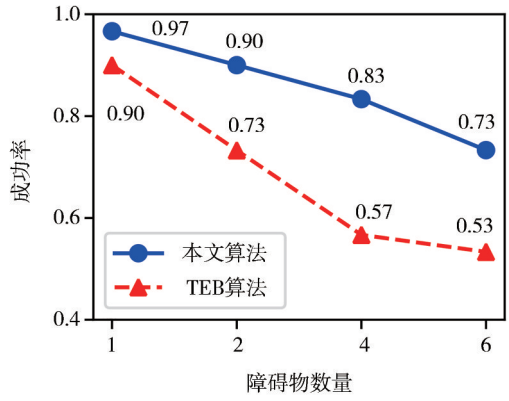


图 11 动态障碍物数量对避障成功率的影响比较

4.2　算法实时性比较

为对算法进行更全面的评价,本文在基于图6所示的仿真动态环境和图8所示的单障碍物场景下,对本文算法和TEB算法的计算耗时进行比较和分析。TEB算法的实现采用了ROS仓库中的官方功能包。两种算法均在Intel i7-10700F/16 GB RAM的计算机平台上运行。在每个场景下,两种算法各进行了100次规划,并统计了TEB算法和本文算法的平均耗时情况,具体数据如表2所示。

表2　仿真动态场景和单障碍物场景下的耗时比较

场景	评价过程	平均耗时/ms
仿真动态场景	TEB算法全过程	23
	本文算法初始轨迹搜索过程	8
	本文算法轨迹优化过程	50
	本文算法全过程	58
单障碍物场景	TEB算法全过程	10
	本文算法初始轨迹搜索过程	5
	本文算法轨迹优化过程	36
	本文算法全过程	41

考虑到本文算法包含初始轨迹搜索和轨迹优化两个主要部分,而TEB算法则直接通过优化过程获得最终轨迹。为更全面地比较和评价算法,对本文算法的初始轨迹搜索过程和轨迹优化过程的平均耗时也进行了统计。本文算法相较于TEB算法在整个过程的耗时上有所增加,但是在动态环境下,本文算法能够提供更高的行驶安全性和更合理的行驶轨迹。此外,需要指出的是,ROS仓库中的TEB算法已经进行了较好的工程优化,例如对动态障碍物的预测处理和轨迹优化过程都采用了多线程技术,而本文算法的所有过程都在单线程条件下运行。如果将本文算法使用多线程技术进行进一步优化,可以在一定程度上减少一些计算耗时。通过对本文算法的各过程耗时进行分析,可以发现主要耗时集中在轨迹优化过程。本文的轨迹优化过程采用了CasADi工具包[19]及其C++API进行建模,并在此框架下调用Ipopt非线性求解器[20]进行求解。若能够直接调用相关的非线性求解器来解决优化问题,将有助于进一步降低本文算法的计算时间。

考虑到剪枝网格的大小对本文算法的计算耗时也会产生影响,因此在动态仿真环境下分别对3种不同尺寸的剪枝网格进行了100次规划,统计结果如表3所示。

随着剪枝网格边长的减小,搜索过程的平均耗时略有增加,而优化过程的平均耗时几乎保持不变。综合考虑各尺寸剪枝网格下的最小耗时和全过程耗时,可发现本文算法的主要耗时集中在优化过程,而剪枝网格的大小对整体耗时影响不大。

表3　仿真动态场景下耗时分析

剪枝网格边长/m	耗时	搜索过程耗时/ms	优化过程耗时/ms	全过程耗时/ms
0.4	平均耗时	4	52	56
	最大耗时	101	136	178
	最小耗时	1	20	21
0.2	平均耗时	8	50	58
	最大耗时	101	86	156
	最小耗时	1	26	27
0.1	平均耗时	14	53	67
	最大耗时	65	119	132
	最小耗时	3	27	30

4.3　实车试验

为进一步验证本文算法在实际应用中的可行性,将本文轨迹规划算法移植到图12的智能车辆平台上进行试验。该智能车配备线控油门、制动和转向系统,并搭载16线激光雷达以感知周围环境。在试验过程中,规划算法所需的二维栅格地图同样通过costmap_2d功能包对激光雷达数据进行实时处理和构建。障碍物的位置信息按照文献[21]中介绍的激光雷达快速聚类方式实时获取,运动障碍物的速度信息由卡尔曼滤波器进行估计。由于试验旨在验证局部轨迹规划的效果,智能车辆的行驶距离较短,所以选择了里程计坐标系作为整个规划系统的基准坐标系。该里程计的定位信息是通过轮式里程计和IMU融合滤波得到的。智能车辆及试验相关参数如表4所示。

实际的场地试验场景如图12所示,设置了锥桶作为静态障碍物,同时另一辆车通过遥控行驶,充当

表4　实车试验相关参数

参数	数值
车长/m	1.92
轴距L/m	1.33
车宽/m	1.02
单个栅格边长/m	0.20
剪枝网格边长/m	0.20
拓展步长τ/s	0.8
速度范围/(m·s^{-1})	[-1.5, 1.5]
最大搜索时间/ms	100
离散加速度集合/(m·s^{-2})	{-0.5, -0.25, 0.0, 0.25, 0.5}
离散前轮转角集合/(°)	{-28, -21, -14, -7, 0, 7, 14, 21, 28}

移动障碍物,虚线箭头为移动障碍物的行驶方向路线。图 13(a)~图 13(d)展示了在该试验场景下的具体避障过程。试验中成功躲避了静态和动态障碍物,验证了本文算法在实际应用中是可行性的。其中,图(a)为试验开始状态,此时车辆正前方存在一个静态障碍物;图(b)为静态障碍物的避障状态,车辆通过右转向成功对静态障碍物进行躲避;图(c)为动态障碍物的避障状态,此时车辆前方出现一移动障碍物,车辆选择右转向然后从移动障碍物后方绕行;图(d)为车辆对动态障碍物躲避成功的状态。

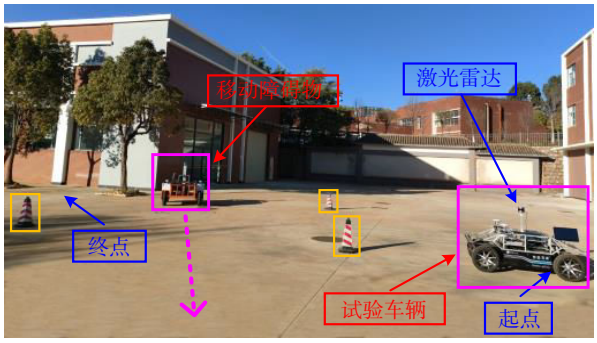


图 12 场地试验场景

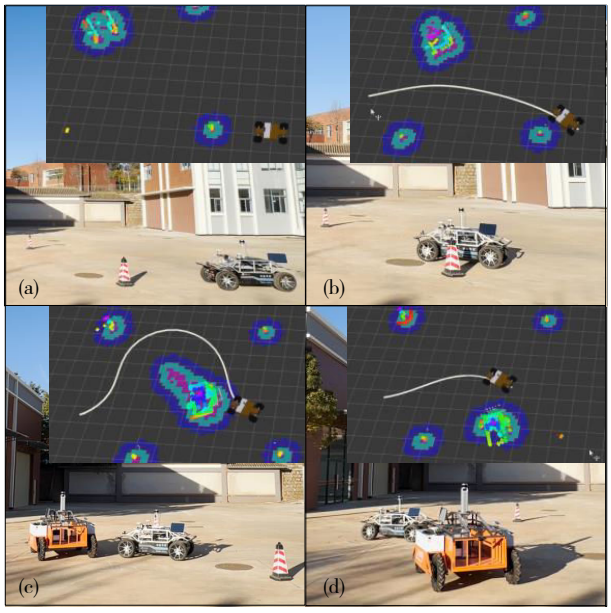


图 13 避障过程展示图

5 结论

针对动态非结构环境下智能车辆轨迹规划问题,提出了一种基于图搜索和优化于一体的轨迹规划方法。通过对智能车辆运动学模型进行离散化和前向模拟获得运动基元,并基于 A*搜索方法对这些运动基元进行探索,得到初始轨迹;然后,以初始轨迹为参考,基于非线性模型预测控制方法对其进一步优化,产生更加平滑和安全的轨迹。此外,针对动态非结构环境提出一种碰撞检测方式,可快速安全地加速基元探索过程。仿真和场地试验结果表明,本文提出的轨迹规划方法在动态非结构环境下所规划的轨迹更为合理,体现出了更高的安全避障能力和可行性,其平均避障成功率达到 85.8%。未来将在规划算法中考虑运动障碍物的运动不确定性,以增强规划算法在动态环境中的鲁棒性。

参考文献

[1] PADEN B, CAP M, YONG S Z, et al. A survey of motion planning and control techniques for self-driving urban vehicles[J]. IEEE Transactions on Intelligent Vehicles, 2016, 1(1): 33-55.

[2] CHIANG H T, HOMCHAUDHURI B, SMITH L, et al. Safety, challenges, and performance of motion planners in dynamic environments[C]// AMATO N M, HAGER G, THOMAS S, et al. Robotics research. Cham: Springer International Publishing, 2020: 793-808.

[3] WANG H, HUANG Y, KHAJEPOUR A, et al. Crash mitigation in motion planning for autonomous vehicles[J]. IEEE Transactions on Intelligent Transportation Systems, 2019, 20(9): 3313-3323.

[4] 田洪清, 丁峰, 郑讯佳, 等. 基于势能场虚拟力的智能网联车辆运动规划[J]. 汽车工程, 2021, 43(4): 518-526.
TIAN H Q, DING F, ZHENG X J, et al. Motion planning based on virtual force of potential field for intelligent connected vehicles[J]. Automotive Engineering, 2021, 43(4): 518-526.

[5] KIM J, KIM K, JO K. Obstacle avoidance in dynamic environment using particle swarm optimization and Kalman filter[C]. 2023 23rd International Conference on Control, Automation and Systems (ICCAS), 2023: 1199-1203.

[6] 郝琨, 张慧杰, 李志圣, 等. 基于改进避障策略和双优化蚁群算法的机器人路径规划[J]. 农业机械学报, 2022, 53(8): 303-312.
HAO K, ZHANG H J, LI Z S, et al. Path planning of mobile robot based on improved obstacle avoidance strategy and double optimization ant colony algorithm[J].Transactions of the Chinese Society for Agricultural Machinery,2022,53(8):303-312.

[7] 胡小建, 周琼, 宋旭东, 等. 基于时间窗约束的多人多储位拣选路径优化模型及改进遗传算法应用研究[J]. 计算机集成制造系统, 2022, 28(11): 3354-3364.
HU X J, ZHOU Q, SONG X D, et al. Optimal model of multi-person and multi-location picking path based on time window constraint and application of improved genetic algorithm[J]. Computer Integrated Manufacturing System, 2022, 28(11): 3354-3364.

[8] JIAN Z, YAN Z, LEI X, et al. Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot[C]. 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023: 3679-3685.

[9] 李军, 周伟, 唐爽. 基于自适应拟合的智能车换道避障轨迹规划[J]. 汽车工程, 2023, 45(7): 1174-1183.
LI J, ZHOU W, TANG S. Lane change and obstacle avoidance trajectory planning of intelligent vehicle based on adaptive fitting [J]. Automotive Engineering, 2023, 45(7): 1174-1183.

[10] LI G, ZHANG X, GUO H, et al. Real-time optimal trajectory planning for autonomous driving with collision avoidance using convex optimization [J]. Automotive Innovation, 2023, 6(3): 481-491.

[11] SINTOV A, SHAPIRO A. Time-based RRT algorithm for rendezvous planning of two dynamic systems [C]. 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014: 6745-6750.

[12] XIN L, KONG Y, LI S E, et al. Enable faster and smoother spatio-temporal trajectory planning for autonomous vehicles in constrained dynamic environment [J]. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 2021, 235(4): 1101-1112.

[13] 齐尧, 朱彦齐, 李永乐, 等. 面向动静态混合环境的智能车运动规划方法[J]. 交通运输系统工程与信息, 2022, 22(4): 293-301.
QI Y, ZHU Y Q, LI Y L, et al. Motion planning method for intelligent vehicles under constrained dynamic and static environment [J]. Journal of Transportation Systems Engineering and Information Technology, 2022, 22(4): 293-301.

[14] 胡杰, 张志豪, 陈瑞楠, 等. 基于改进混合A*的智能汽车时空联合规划方法[J]. 汽车工程, 2023, 45(7): 1123-1133.
HU J, ZHANG Z H, CHEN R N, et al. Spatio-temporal joint planning method of intelligent vehicles based on improved hybrid A*[J]. Automotive Engineering, 2023, 45(7): 1123-1133.

[15] PETTI S, FRAICHARD T. Safe motion planning in dynamic environments[C]. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2005: 2210-2215.

[16] AMANATIDES J, WOO A. A fast voxel traversal algorithm for ray tracing[C]. Eurographics, 1987, 87(3): 3-10.

[17] BAUTIN A, MARTINEZ-GOMEZ L, FRAICHARD T. Inevitable collision states: a probabilistic perspective[C]. 2010 IEEE International Conference on Robotics and Automation, 2010: 4022-4027.

[18] LIN J, ZHOU T, ZHU D, et al. Search-based online trajectory planning for car-like robots in highly dynamic environments[C]. 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021: 8151-8157.

[19] ANDERSSON J A E, GILLIS J, HORN G, et al. CasADi-a software framework for nonlinear optimization and optimal control[J]. Mathematical Programming Computation, 2019, 11(1): 1-36.

[20] WÄCHTER A, BIEGLER L T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming [J]. Mathematical Programming, 2006, 106: 25-57.

[21] YAN Z, DUCKETT T, BELLOTTO N. Online learning for 3D LiDAR-based human detection: experimental analysis of point cloud clustering and classification methods[J]. Autonomous Robots, 2020, 44(2): 147-164.