控制与决策
Control and Decision

运动动力学约束下基于自适应参数的运动规划方法

崔冰, 李广, 胡飞扬, 高寒, 夏元清

引用本文:
崔冰, 李广, 胡飞扬, 等. 运动动力学约束下基于自适应参数的运动规划方法[J]. 控制与决策, 2025, 40(5): 1660-1668.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2024.0676

您可能感兴趣的其他文章

Articles you may be interested in

基于Frenet坐标系的自动驾驶轨迹规划与优化算法

Trajectory planning and optimization algorithm for automated driving based on Frenet coordinate system

控制与决策. 2021, 36(4): 815-824 https://doi.org/10.13195/j.kzyjc.2019.0748

有限频域线性重复过程的动态迭代学习控制

Dynamic iterative learning control for linear repetitive processes over finite frequency ranges

控制与决策. 2021, 36(3): 599-608 https://doi.org/10.13195/j.kzyjc.2019.0873

四旋翼无人机抗干扰轨迹跟踪控制

Anti-interference trajectory tracking control of quadrotor UAV

控制与决策. 2021, 36(2): 379-386 https://doi.org/10.13195/j.kzyjc.2019.0875

基于改进卷积神经网络的动力下肢假肢运动意图识别

Intent recognition of power lower-limb prosthesis based on improved convolutional neural network

控制与决策. 2021, 36(12): 3031-3038 https://doi.org/10.13195/j.kzyjc.2020.0326

基于神经动态优化的非线性系统近似最优跟踪控制

Approximate optimal tracking control for nonlinear systems based on neurodynamic optimization

控制与决策. 2021, 36(1): 97-104 https://doi.org/10.13195/j.kzyjc.2020.0056

运动动力学约束下基于自适应参数的运动规划方法

崔 冰†，李 广，胡飞扬，高 寒，夏元清

(北京理工大学 自动化学院，北京 100081)

摘 要: 稳定稀疏探索树 (SST) 是一种基于采样的渐近最优运动规划算法, 与传统的渐近最优算法 RRT*相比, SST 采用随机前向传播来生成新节点, 无需求解两点边值问题 (BVP), 即可直接规划出一条满足机器人运动学和动力学约束的可行轨迹. 针对 SST 对参数敏感、难以适应复杂多变的环境等问题, 提出一种基于自适应参数的 SST 算法 (ASST), 利用规划过程中的节点碰撞率和节点密度等已知信息, 对节点所处的环境区域和邻居信息进行估计, 自适应地改变节点选择半径和节点剪枝半径. 最后, 对多种系统动态和复杂环境类型进行了仿真验证, 仿真结果表明该算法能降低对参数的依赖性, 在复杂困难环境中能够提高求解成功率和计算效率, 对不同规划问题具有较强的适应性.

关键词: 运动规划；随机前向传播；运动动力学约束；自适应参数；稳定稀疏探索树

中图分类号: TP273 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2024.0676

引用格式: 崔冰, 李广, 胡飞扬, 等. 运动动力学约束下基于自适应参数的运动规划方法 [J]. 控制与决策, 2025, 40(5): 1660-1668.

Motion planning based on adaptive parameters under kinodynamic constraints

CUI Bing†，*LI Guang*，*HU Fei-yang*，*GAO Han*，*XIA Yuan-qing*

(School of Automation，Beijing Institute of Technology，Beijing 100081，China)

Abstract: Stable sparse RRT(SST) is a sampling-based asymptotically optimal motion planning algorithm. Compared with the traditional asymptotically optimal algorithm RRT*, the SST employs random forward propagation to generate new nodes, without solving the two-point boundary value problem(BVP), and can directly plan a feasible trajectory that satisfies the system's kinodynamic constraints. Considering the issues associated with SST's sensitivity to parameters and challenges in adapting to complex and dynamic environments, an improved SST algorithm with adaptive parameters(ASST) is proposed. By utilizing known information such as node collision rate and node density during the planning process, the environmental area and neighborhood information of the node are estimated, and then the node selection radius and node pruning radius are adaptively changed. Simulation experiments have evaluated various types of system dynamics and complex environments, and the experimental results show that the proposed algorithm can reduce the dependence on parameters, improve the success rate and computational efficiency in complex environments, and have strong adaptability to different motion planning problems.

Keywords: motion planning；random forward propagation；kinodynamic constraints；adaptive parameters；SST

0 引 言

基于采样的运动规划方法, 如概率路线图 (PRM)[1] 和快速探索随机树 (RRT)[2] 等, 因其计算效率高、通用性好、易于实现等优点, 已被广泛应用于解决高维状态空间的运动规划问题[3-7]. 然而, 这种方法没有考虑实际系统的运动学和动力学约束, 只能规划出一条无碰撞的几何路径, 通常在实际场景中无法直接执行[8]. 因此, 考虑运动学和动力学约束的运动动力学运动规划 (MP) 得到了广泛的研究[9-11]. 在 KMP 问题中, 最具挑战性的难题是在动力学约束下设计连接任意两个中间状态的局部转向函数[12]. 这实际上是一个两点边值问题 (BVP), 通常无法直接求得其解析解[13].

针对以上问题, 研究者们提出了一类不需要求

收稿日期: 2024-06-06；**录用日期:** 2024-10-12.
基金项目: 国家自然科学基金项目 (62173041).
责任编委: 关新平.
†通信作者. E-mail: bing.cui@bit.edu.cn.

解两点边值问题的随机前向传播算法来解决 KMP 问题[14-17]. 这类方法对控制空间随机采样, 将采样得到的控制作用于一个选定的节点, 使其向前传播得到新的节点, 并通过剪枝操作修剪掉一些无意义的节点以降低树节点数量, 提高计算效率. 这类方法已被证明具有概率完备性[18] 和渐近最优性[19], 对于不同的系统动态均表现出良好的适用性. 其中最具代表性的算法是稳定稀疏树 (SST)[14]. SST 算法通过引入δ_{BN}和δ_s两个参数, 分别控制节点选择和节点修剪的半径范围, 优先选择代价较小的节点进行随机前向传播, 并对传播得到的新节点进行剪枝. 这一过程实现了高质量路径的有效收敛, 同时保持了数据结构的稀疏性[20]. 对控制空间的随机采样虽然有助于解决 BVP 问题, 但同时也导致 SST 的计算效率显著降低. 为解决这一问题, 研究者们提出一系列改进算法. 文献 [21] 利用启发式信息指导节点的选择, 并引入 Blossom 算法[22], 从多个候选控制中选择最优的控制动作进行前向传播, 提高求解质量和计算效率. 文献 [23] 利用 VAE-GAN 模型生成控制动作, 减少不必要的节点扩展, 降低 SST 的内存占用, 提高了算法的求解速度. 文献 [24] 在 SST 中使用极值控制作为输入, 提高了算法的收敛速度并给出了理论证明.

尽管一系列改进提高了 SST 算法的计算效率, 但 SST 算法仍面临着参数较多、难以调参的问题. 为解决此问题, 文献 [25] 引入支配性信息区域 (DIR) 表示配置空间的探索情况, 其大小反映区域的探索稀疏度. 基于 DIR 的算法能够有效平衡探索与利用之间的权衡. 与原 SST 算法相比, 新算法将参数减少至一个, 降低了对参数的依赖, 但仍需多次调整以获得最佳性能. 此外, 该算法仅基于探索情况进行求解, 而未考虑障碍物对探索过程的影响, 导致其在复杂环境中的适用性受到限制.

综合已有研究成果及存在的问题, 本文提出了一种自适应参数的改进 SST 算法 (ASST). 针对传统 SST 算法对参数过于依赖且难以调参的问题, ASST 利用树的扩展过程提供的已知信息, 自适应地改变节点选择参数δ_{BN}和节点剪枝参数δ_s, 避免了繁琐的调参过程, 提高了算法在复杂环境中的适用性, 对各种规划问题具有较强的适应性. 在不同的环境和动态系统下对所提出的 ASST 算法进行了评估, 仿真结果表明, 在绝大多数情况下, ASST 算法相较于 SST 均有更好的表现, 验证了该算法的有效性.

1　SST 算法基本原理

SST 是一种基于增量采样的渐近最优算法, 绕过了 BVP 问题的求解, 被广泛用于求解复杂动力学约束下的 KMP 问题. SST 算法主要包括节点选择, 节点传播, 节点剪枝 3 个部分. 具体流程如算法 1 所示.

Algorithm 1　SST($\mathbb{X}, \mathbb{U}, x_0, T_{\text{prop}}, N, \delta_{BN}, \delta_s$).

1) $\mathbb{V}_{\text{active}} \leftarrow \{x_0\}, \mathbb{V}_{\text{inactive}} \leftarrow \emptyset$;
2) $G = \{V \leftarrow (\mathbb{V}_{\text{active}} \bigcup \mathbb{V}_{\text{inactive}}), \mathbb{E} \leftarrow \emptyset\}$;
3) $s_0 \leftarrow x_0, s_0.\text{rep} = x_0, S \leftarrow \{s_0\}$;
4) **for** N iterations **do**
5)　$x_{\text{selected}} \leftarrow$ Best_First_Select($\mathbb{X}$, $\mathbb{V}_{\text{active}}$, δ_{BN});
6)　$x_{\text{new}} \leftarrow$ MonteCarlo-Prop($x_{\text{selected}}, \mathbb{U}, T_{\text{prop}}$);
7)　**if** CollisionFree($\overrightarrow{x_{\text{selected}} \rightarrow x_{\text{new}}}$) **then**
8)　　**if** Locally_Best($x_{\text{new}}, S, \delta_s$) **then**
9)　　$\mathbb{V}_{\text{active}} \leftarrow \mathbb{V}_{\text{active}} \bigcup \{x_{\text{new}}\}$;
10)　　$\mathbb{E} \leftarrow \mathbb{E} \bigcup \{\overrightarrow{x_{\text{selected}} \rightarrow x_{\text{new}}}\}$;
11)　　Pruning($x_{\text{new}}, \mathbb{V}_{\text{active}}, \mathbb{V}_{\text{inactive}}, \mathbb{E}$);
12)　　**end if**
13)　**end if**
14) **end for**
15) **return** G.

在节点选择过程中, SST 算法采用 Best_First_Select 方法选择将被扩展的节点. 该方法在一个δ_{BN}为半径的圆内找到一组邻居节点, 并选择其中代价最低的节点x_{selected}进行扩展, 平衡了节点探索与利用的权衡, 提高了求解的质量.

在节点传播过程中, SST 算法采用 MonteCarlo-Prop 方法来扩展被选择的节点. 该方法随机采样控制动作和持续时间, 对被选择的节点进行前向传播以获得新的节点, 绕过了 BVP 问题的求解.

在节点剪枝过程中, 不再是局部最优的节点将被剪枝以降低节点数量. 为判断新节点是否为局部最优, 需要定义局部邻域. SST 中, 被称为“目击者”的节点s主导着以自身为圆心、以δ_s为半径的局部邻域. 每当扩展得到的新节点x_{new}不在任何一个目击者s的主导范围内时, 则将该新节点作为新的目击者加入到目击者集S中. 对于集合S中的每一个目击者s, 在树节点中都有且仅有一个节点可以代表该目击者, 且在局部邻域内代价最低, 这个节点被称为目击者的代表$s.\text{rep}$, 是局部邻域内的主导节点.

SST 算法是一种渐近最优算法, 无需 BVP 求解器即可求解运动规划问题, 且能保持较为稀疏的节点分布. 但 SST 算法的性能较为依赖于参数δ_s和δ_{BN}

的选择, 无法适应复杂多变的环境.

2 ASST 算法设计

针对 SST 算法存在的问题, 提出一种基于自适应参数的改进 SST 算法 (ASST). 该算法对参数δ_s和δ_{BN}做了自适应设计, 能显著降低算法对参数的依赖, 提升对复杂环境的适应性, 提高求解速度和规划成功率.

2.1 自适应δ_s

在节点剪枝过程中, δ_s作为剪枝半径, 控制着剪枝范围的大小. 增大δ_s有助于执行更多的剪枝操作, 保存更少的节点以减少计算负担, 但如果δ_s过大将导致在狭窄通道内的采样可能无法被保存, 降低运动规划的求解成功率.

为减少对δ_s的依赖, 提高通过狭窄通道的成功率和计算效率, 本节提出一种自适应δ_s算法, 引入节点碰撞率cr表示当前节点发生碰撞的频率, 根据cr值来自适应地改变δ_s的值. 该算法使得树节点在无碰撞的自由区域分布更稀疏, 而在障碍物和复杂环境区域分布更密集. 自适应参数

$$\delta_s=\begin{cases}\delta_{s_0}+\gamma(\delta_{s\_\max}-\delta_{s\_\min}),\ \mathrm{cr}=0;\\ \delta_{s_0}(1-\mathrm{cr}),\ \mathrm{cr}\neq 0.\end{cases}\tag{1}$$

其中: $\delta_{s_\max}$和$\delta_{s_\min}$分别表示自适应δ_s的最大值和最小值; $\gamma<1$是非负扩展系数, 表示参数的扩展幅度; δ_{s_0}表示当前节点从父节点处继承的初值, 其计算过程如算法 2 所示. 首先在S集合中找到最接近x_{selected}的目击者, 记为s_{nearest}. 若新节点x_{new}和s_{nearest}的距离大于x_{selected}的δ_s, 则将x_{selected}的δ_s扩大并赋值给x_{new}的δ_{s_0}; 如果x_{new}在s_{nearest}的主导范围内且局部最优, 则目击者的代表s_{nearest}.rep被新节点代替, x_{new}将继承被替换节点的δ_s.

Algorithm 2 Init_Delta($x_{\text{new}}, x_{\text{selected}}, S$).

1) $s_{\text{nearest}} \leftarrow \text{nearest}(S, x_{\text{new}})$;

2) **if** $\|x_{\text{new}}-s_{\text{nearest}}\|>x_{\text{selected}}.\delta_s$ **then**

3) $\quad x_{\text{new}}.\delta_{s_0}\leftarrow x_{\text{selected}}.\delta_s+\gamma(\delta_{s\_\max}-\delta_{s\_\min})$;

4) **else**

5) $\quad x_{\text{new}}.\delta_{s_0}\leftarrow s_{\text{nearest}}.\text{rep}.\delta_s$;

6) **end if**

7) $x_{\text{new}}.\delta_{s_0}\leftarrow \min(\max(x_{\text{new}}.\delta_{s_0},\delta_{s\_\min}),\delta_{s\_\max})$;

8) **return** $x_{\text{new}}.\delta_{s_0}$.

在得到节点初值δ_{s_0}后, 根据δ_s的自适应公式 (1) 可计算当前节点的自适应δ_s值, 过程如算法 3 所示. 该算法实现了δ_s的自适应变化, 当cr较大, 即当前节点靠近障碍物时, δ_s值会自适应地减小以提高通过复杂环境的成功率; 而cr较小时, 表明当前节点处在自由宽阔的无障碍区域, δ_s会自适应地扩大以维持稀疏的数据结构, 减少计算负担.

Algorithm 3 Update_Delta(x_{selected}).

1) **if** $x_{\text{selected}}.\mathrm{cr}=0$ **then**

2) $\quad x_{\text{selected}}.\delta_s\leftarrow x_{\text{selected}}.\delta_{s_0}+\gamma(\delta_{s\_\max}-\delta_{s\_\min})$;

3) **else**

4) $\quad x_{\text{selected}}.\delta_s\leftarrow x_{\text{selected}}.\delta_{s_0}(1-x_{\text{selected}}.\mathrm{cr})$;

5) **end if**

6) $x_{\text{selected}}.\delta_s\leftarrow \min(\max(x_{\text{selected}}.\delta_{s_0},\delta_{s\_\min}),\delta_{s\_\max})$;

7) **return** $x_{\text{selected}}.\delta_s$.

在本节引入的节点碰撞率cr是规划过程中的已知信息, 表示当前节点的碰撞频率, 可反映在给定环境下当前节点传播到新节点的难度. 若cr较大, 则代表该节点在随机控制的作用下发生碰撞的次数较多, 表示当前节点处于复杂环境中. 虽然基于已知信息的碰撞率cr并不等同于准确的碰撞概率, 但能一定程度上反映当前节点的碰撞趋势, 可作为剪枝半径δ_s自适应变化的依据. 节点碰撞率

$$\mathrm{cr}=\frac{c}{p}m(p).\tag{2}$$

其中: c为当前节点的碰撞次数; p为当前节点的传播次数; $m(p)$是与p相关的修正函数, 需满足以下条件:

$$m'(p)>0,\tag{3}$$

$$\lim_{p\to\infty}m(p)=1,\tag{4}$$

$$\frac{c}{p+\Delta}m(p+\Delta)<\frac{c}{p}m(p)<\frac{c+\Delta}{p+\Delta}m(p+\Delta),\tag{5}$$

Δ是新增的传播次数. 在本文中修正函数选取$m(p)=\dfrac{p}{p+1}$. 修正函数$m(p)$可以在当前节点传播次数较少时, 减缓cr的快速变化, 提高cr表示碰撞概率的置信程度. 随着p的增加, 碰撞率cr会变得更加精确, 因此修正函数$m(p)$应满足式 (3) 和 (4). 而修正函数也不应改变原有节点碰撞率的大小关系, 因此修正函数$m(p)$还需满足式 (5).

2.2 自适应δ_{BN}

在节点选择过程中, δ_{BN}作为选择半径, 对选择范围的大小起着决定性作用. 较大的δ_{BN}值意味着求解获得的初始轨迹质量更高, 同时也伴随着更大的

计算负担. 然而, 如果δ_{BN}取值过大, 则可能会导致对状态空间的探索不足, 靠近根节点的节点可能被重复选择. 这不仅会影响求解时间, 还可能导致求解失败. 基于上述原因, 本节提出一种关于δ_{BN}的自适应算法, 利用区域间的节点密度来自适应地改变δ_{BN}的值, 使得在节点密度低的区域更注重求解质量, 在节点密度较大的区域能保持较小计算量, 并避免重复选择靠近根的节点, 提高求解成功率.

自适应参数

$$\delta_{BN}=\delta_{BN\_\text{free}}\left(1-\frac{\sum_{i=1}^{n}\text{cr}_i}{n}\right). \tag{6}$$

其中: cr_i是第i个节点的节点碰撞率; δ_{BN_free}为无障碍区域中自适应调节后的δ_{BN}值, 有

$$\delta_{BN\_\text{free}}=\begin{cases}\delta_{BN\_\max}, & n\leqslant n_s;\\ \delta_{BN\_\max}\sqrt[d]{\dfrac{n_s}{n}}, & n_s<n\leqslant\left(\dfrac{\delta_{BN\_\max}}{\delta_{BN\_\min}}\right)^d n_s;\\ \delta_{BN\_\min}, & \text{otherwise}.\end{cases} \tag{7}$$

其中: $\delta_{BN_\max}$和$\delta_{BN_\min}$分别是自适应δ_{BN}取值范围的最大最小值; d表示空间维度; n是当前节点在$\delta_{BN_\max}$半径范围内的邻居节点数;n_s是算法设置的处在δ_{BN_free}内的期望邻居节点数; δ_{BN_free}值是由$\delta_{BN_\max}$根据当前节点密度 (即$\delta_{BN_\max}$范围内的邻居节点数n) 收缩得到的, 其目标是将δ_{BN_free}调节到一个合理的值, 使得处在其中的节点数等于n_s.

式 (6) 中的$1-\sum_{i=1}^{n}cr_i\big/n$对障碍物区域做了相应的修正, 其中$\sum_{i=1}^{n}cr_i\big/n$表示当前节点$\delta_{BN_\max}$半径范围内的平均碰撞率. 考虑到障碍物会占用节点空间, 实际的节点密度和无障碍区域的节点密度存在着偏差. 如图 1 所示, 蓝叉表示障碍物中的节点. 在无障碍空间中, 蓝叉本可以成为树节点, 但实际上由于处在障碍物内, 这些节点将不会加入到树节点中, 使得$\delta_{BN_\max}$半径内的节点数比无障碍区域更少, 导致自适应δ_{BN}的收缩较小. 因此, 将自适应δ_{BN}与反映节点所在空间区域的节点碰撞率cr相结合, 节点碰撞率越大, 意味着节点附近存在障碍的概率越大, 更需要加大δ_{BN}的收缩. 修正后的自适应δ_{BN}相比δ_{BN_free}有着更小的值, 在复杂障碍物中能更有效地找到关键节点 (即图中$x_{\text{selected_new}}$), 提高通过狭窄通道和复杂环境的成功率.

基于上述的自适应参数δ_{BN}, 算法 4 对传统

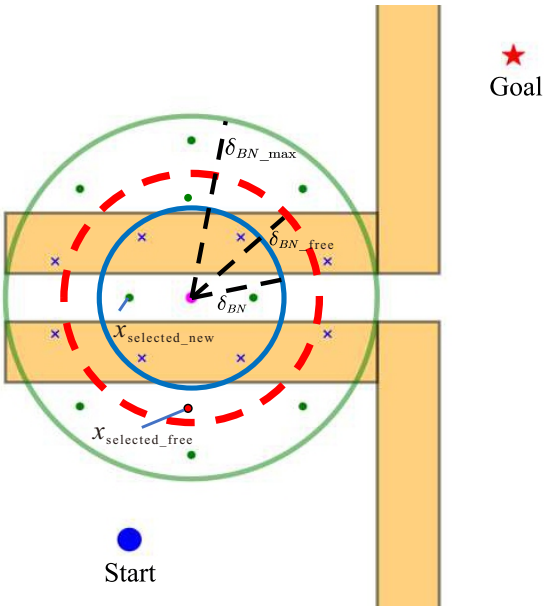


图1　自适应参数δ_{BN}的修正

SST 算法的节点选择过程做了改进, 主要区别在于算法 4 使用自适应δ_{BN}来查找x_{rand}附近的一组状态, 从而更好地平衡计算负担与求解质量. 该算法首先使用$\delta_{BN_\max}$查找所有可能的邻居节点的集合X_{near} (第 2 步), 并在X_{near}不为空时, 根据邻居节点的数量计算自适应δ_{BN}(第 3 步～ 第 7 步). 然后, 以x_{rand}为中心, 计算得到的自适应δ_{BN}为半径, 查找在集合X_{near}中的真正邻居节点子集X_{adapt}(第 8 步). 如果X_{adapt}为空集, 则程序将返回$\mathbb{V}$中最近的节点, 否则将返回集合X_{adapt}中代价最低的节点 (第 9 步 ～ 第 12 步).

Algorithm 4　Adaptive_Selection($\mathbb{X}$, $\mathbb{V}$).

1) $x_{\text{rand}}\leftarrow$ Sample_State($\mathbb{X}$);
2) $X_{\text{near}}\leftarrow$ Near($\mathbb{V}, x_{\text{rand}}, \delta_{BN\_\max}$);
3) **if** $X_{\text{near}}=\emptyset$ **then**
4)　　**return** Nearest($\mathbb{V}, x_{\text{rand}}$);
5) **else**
6)　　$n\leftarrow$ Get_Nodes_Number(X_{near});
7)　　$\delta_{BN}\leftarrow$ Calculate_Radius(n, n_s, X_{near});
8)　　$X_{\text{adapt}}\leftarrow$ Near($X_{\text{near}}, x_{\text{rand}}, \delta_{BN}$);
9)　　**if** $X_{\text{adapt}}=\emptyset$ **then**
10)　　**return** Nearest($\mathbb{V}, x_{\text{rand}}$);
11)　　**end if**
12)　　**return** $\arg\min_{x\in X_{\text{adapt}}}\text{cost}(x)$;
13) **end if**

2.3 ASST 算法

根据前文对参数δ_{BN}和δ_s的自适应设计, 本节提出使用自适应参数的 ASST 算法, 详见算法 5. 算法 5 将 SST 中 Best_First_Selection 算法替换为 Adaptive_Selection 算法, 以使用自适应δ_{BN}; 用$x_{\text{selected}}.\delta_s$替换固定$\delta_s$参数, 以使用自适应$\delta_s$. 如果一个新节点是局部最佳的, 则 Init_Delta 算法将为此新节点计算初始的δ_s, 并将x_{new}的碰撞率cr设置为0, x_{new}将主导其局部邻域, 而被替换的节点将被剪枝. 在 CollisionFree 过程结束后, 所选节点将根据碰撞情况更新cr和δ_s.

Algorithm 5 Adaptive SST($\mathbb{X}$, $\mathbb{U}$, x_0, T_{prop}, N).

1) $\mathbb{V}_{\text{active}} \leftarrow \{x_0\}, \mathbb{V}_{\text{inactive}} \leftarrow \emptyset$;

2) $G = \{V \leftarrow (\mathbb{V}_{\text{active}} \bigcup \mathbb{V}_{\text{inactive}}), \mathbb{E} \leftarrow \emptyset\}$;

3) $x_0.\text{cr} \leftarrow 0, s_0 \leftarrow x_0, s_0.\text{rep} = x_0, S \leftarrow \{s_0\}$;

4) **for** N iterations **do**

5) $x_{\text{selected}} \leftarrow$ Adaptive_Selection($\mathbb{X}$, $\mathbb{V}_{\text{active}}$);

6) $x_{\text{new}} \leftarrow$ MonteCarlo-Prop($x_{\text{selected}}, \mathbb{U}, T_{\text{prop}}$);

7) **if** CollisionFree($\overrightarrow{x_{\text{selected}} \to x_{\text{new}}}$) **then**

8) **if** Locally_Best($x_{\text{new}}, S, x_{\text{selected}}.\delta_s$) **then**

9) $x_{\text{new}}.\delta_{s_0} \leftarrow$ Init_Delta(x_{new}, x_{selected}, S);

10) $x_{\text{new}}.\delta_s \leftarrow x_{\text{new}}.\delta_{s_0}$;

11) $x_{\text{new}}.\text{cr} \leftarrow 0$;

12) $\mathbb{V}_{\text{active}} \leftarrow \mathbb{V}_{\text{active}} \bigcup \{x_{\text{new}}\}$;

13) $\mathbb{E} \leftarrow \mathbb{E} \bigcup \{\overrightarrow{x_{\text{selected}} \to x_{\text{new}}}\}$;

14) Pruning($x_{\text{new}}, \mathbb{V}_{\text{active}}, \mathbb{V}_{\text{inactive}}, \mathbb{E}$);

15) **end if**

16) **end if**

17) $x_{\text{selected}}.\text{cr} \leftarrow$ Update_cr(x_{selected});

18) $x_{\text{selected}}.\delta_s \leftarrow$ Update_Delta(x_{selected});

19) **end for**

20) **return** G.

3 仿真验证和结果分析

为了验证本文所提出方法的有效性, 本节总共设置了 4 个仿真以对比不同参数下的 SST 算法和 ASST 算法的规划性能. 所有仿真均在同一平台上进行, 该平台采用 16 GB RAM 和 3.20 GHz × 8 AMD Ryzen 7 5800H 处理器.

3.1 不同动态系统的仿真对比

为了验证 ASST 算法能够在多种动态系统中有效运行, 本节在狭窄通道和杂乱障碍物的混合环境下对 3 个动态系统做了仿真对比, 各系统的运动动力学模型如下:

1) 双积分器系统: 双积分器系统是一个具有完整约束的动力学系统, 具有四维状态空间和二维控制空间. 状态空间包含位置向量$\boldsymbol{p}$和速度向量$\boldsymbol{v}$, 其边界设置为$\boldsymbol{p} \in [0, 150] \times [0, 100]$、$\boldsymbol{v} \in [-5, 5] \times [-5, 5]$, 在控制空间中, 控制输入边界设为$\boldsymbol{u} \in [-10, 10] \times [-10, 10]$, 采样时间$T_{\text{prop}} \in (0, 1.2]$.

2) 无人车系统: 无人车系统采用阿克曼转向模型, 是一个非完整约束系统, 其运动学模型如下:

$$\begin{bmatrix} \dot{x} & \dot{y} & \dot{\varphi} \end{bmatrix}^{\mathrm{T}} = v \begin{bmatrix} \cos\varphi & \sin\varphi & \dfrac{\tan\delta}{L} \end{bmatrix}^{\mathrm{T}}.$$

其中: $[x, y, \varphi]$为位置和方向的状态向量, v为无人车的速度, φ为无人车的航向角, δ为无人车前轮的转向角, L为无人车的轴距. 状态空间的边界设为$[x, y, \varphi] \in [0, 150] \times [0, 100] \times [-\pi, \pi]$, 控制输入边界设为$[v, \delta] \in [0, 10] \times [-\pi/6, \pi/6]$, 轴距设置为$L = 4$. 采样时间$T_{\text{prop}} \in (0, 1.2]$.

3) Cart-Pole 倒立摆系统: Cart-Pole 系统是一个非线性欠驱动的非完整系统, 具有四维状态向量和一维的控制输入, 其动力学系统如下:

$$\boldsymbol{x} = \begin{bmatrix} \dot{\theta} & \dot{p} & \theta & p \end{bmatrix}^{\mathrm{T}},$$

$$\ddot{\theta} = \frac{-3m_2 l\dot{\theta}^2 \sin\theta\cos\theta - 6(m_1+m_2)g\sin\theta}{4l(m_1+m_2) - 3m_2 l\cos^2\theta} - \frac{6(u - b\dot{p})\cos\theta}{4l(m_1+m_2) - 3m_2 l\cos^2\theta},$$

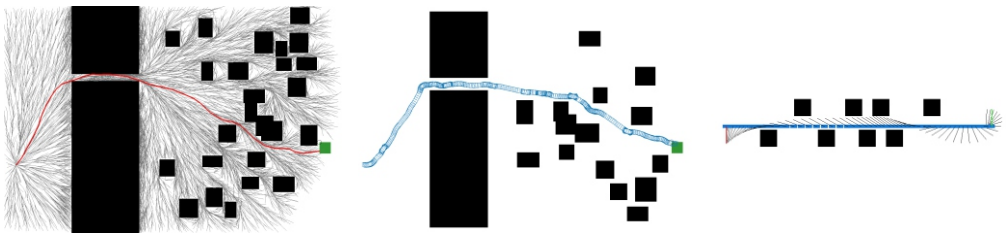
$$\ddot{p} = \frac{2m_2 l\dot{\theta}^2 \sin\theta + 3m_2 g\sin\theta\cos\theta + 4u - 4b\dot{p}}{4(m_1+m_2) - 3m_2 l\cos^2\theta}.$$

其中: m_1表示车的质量, m_2表示杆的质量, l表示杆的长度, θ表示杆摆的角度, p表示车的位置, b表示车与轨道之间的摩擦系数, g表示重力加速度. 以上参数设置为$m_1 = 0.5, m_2 = 0.5, l = 0.5, b = 0.1, g = 9.8$, 状态空间的边界设为$p \in [0, 10]$, 控制输入边界设为$u \in [-20, 20]$. 采样时间$T_{\text{prop}} \in (0, 0.5]$.

仿真环境中 3 种动态系统设置的地图如下: 双积分器系统地图尺寸为 150 m×100 m, 地图包含一个宽度为 30 m、间距为 2 m 的狭窄通道以及通道右侧随机生成的 25 个边长小于 10 m 的障碍物; 无人车地图尺寸为 150 m×100 m, 考虑到无人车本身具有一定的体积, 地图包含一个宽度为 30 m、间距为 4 m 的狭窄通道以及通道右侧随机生成的 15 个边长小于 10 m 的矩形障碍物; Cart-Pole 倒立摆地图则包含上下两侧各自随机生成的 4 个边长为 5 m 的正方形障碍物, 上下两侧的通道间距分别为 3 m 和 1 m. ASST

算法在 3 种系统上的运行结果如图 2 所示.

为了评估所提出的 ASST 算法相较于传统 SST 算法在性能上的优劣, 仿真对 SST 算法设置了 9 组不同的参数组合, 并与所提出的 ASST 算法进行对比分析. SST 算法和 ASST 算法在不同系统的参数设置如表 1 所示, ASST 算法的非负扩展系数和期望邻居节点数根据经验分别选取为$\gamma = 0.1$, $n_s = 8$. 仿真将对初始求解时间和规划成功率两个指标进行评估. 为了获得准确的仿真结果, 每组动态系统在 10 张不同的地图中分别进行 10 次仿真, ASST 算法与 SST 算法的性能指标对比如图 3 所示. 仿真结果表明, 在双积分器系统中, SST5 在各组 SST 参数中表现最优, 其求解时间均值为 11.50 s, 中位数为 7.71 s; 相比之下, ASST 算法的求解时间均值为 9.73 s, 中位数为 6.45 s, 优于 SST5. 在无人车系统中, SST5 在各组 SST 参数中表现最优, 其求解时间均值为 12.07 s, 中位数为 6.68 s; 而 ASST 算法的求解时间均值为 9.80 s, 中位数为 6.03 s, 优于 SST5. 在 Cart-Pole 倒立摆系统中, SST3 在各组 SST 参数中表现最优, 其求解时间均值为 4.56 s, 中位数为 3.11 s; 相比之下, ASST 算法的求解时间均值为 4.19 s, 中位数为 3.18 s, 略优于 SST3. 在规划成功率上, ASST 算法在双积分器系统、无人车系统以及 Cart-Pole 倒立摆系统的规划成功率分别是 0.97, 0.96, 0.98, 均不低于任何一组

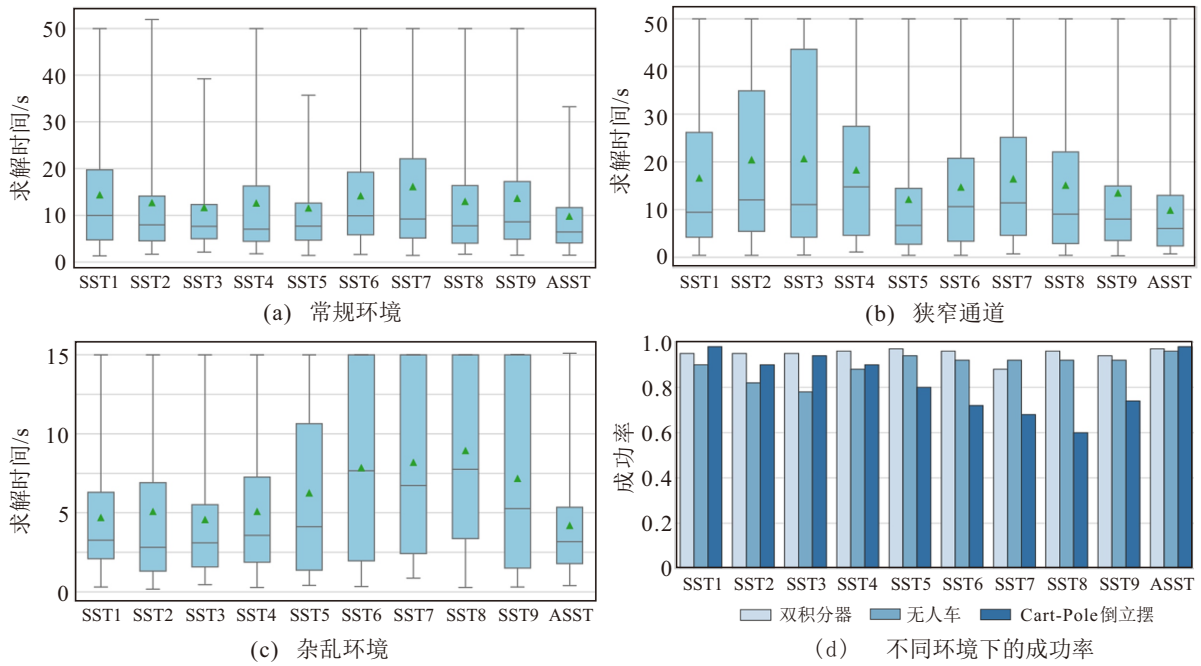


(a) 双积分器系统　　(b) 无人车系统　　(c) Cart-Pole倒立摆系统

图2　ASST 算法在不同系统下的仿真结果

表1　不同系统动态仿真参数设置

	参数设置	SST1	SST2	SST3	SST4	SST5	SST6	SST7	SST8	SST9	ASST
双积分器	δ_{BN}	2	4	8	2	4	8	2	4	8	[2, 8]
	δ_s	0.2	0.2	0.2	1	1	1	2	2	2	[0.2, 2]
无人车	δ_{BN}	2	4	8	2	4	8	2	4	8	[2, 8]
	δ_s	0.2	0.2	0.2	1	1	1	2	2	2	[0.2, 2]
Cart-Pole倒立摆	δ_{BN}	0.6	0.9	1.2	0.6	0.9	1.2	0.6	0.9	1.2	[0.6, 1.2]
	δ_s	0.2	0.2	0.2	0.4	0.4	0.4	0.6	0.6	0.6	[0.2, 0.6]



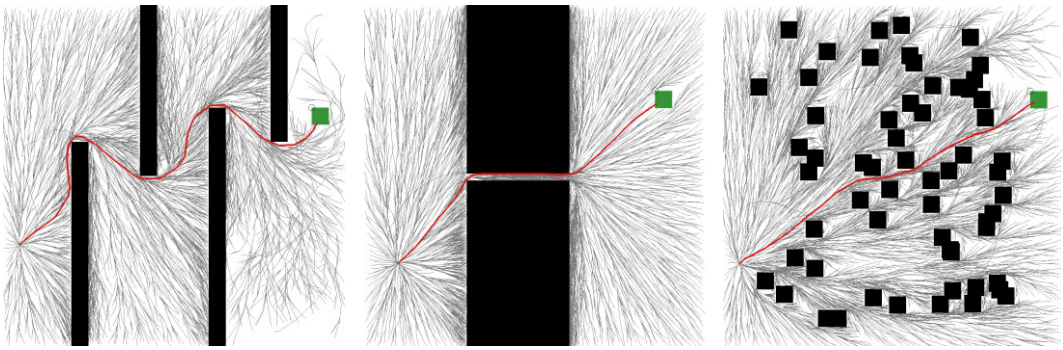
(a) 常规环境　(b) 狭窄通道　(c) 杂乱环境　(d) 不同环境下的成功率

图3　不同动态系统的性能指标对比

SST 参数. 以上结果充分验证了在多种动态系统中, ASST 算法均具有更快的求解速度和更高的规划成功率. 另外, SST 算法在较为复杂的动态系统中 (如无人车系统, Cart-Pole 倒立摆系统), 不同参数的性能差距较大, 对参数的选取较为敏感, 而 ASST 在给定δ_{BN}和δ_s的取值范围内比任何一组参数的 SST 算法都有更好的表现. 综上所述, ASST 算法极大程度地降低了对参数的依赖性, 提高了对复杂多变环境的适应能力, 在多种系统中均有良好表现.

3.2 不同环境的仿真对比

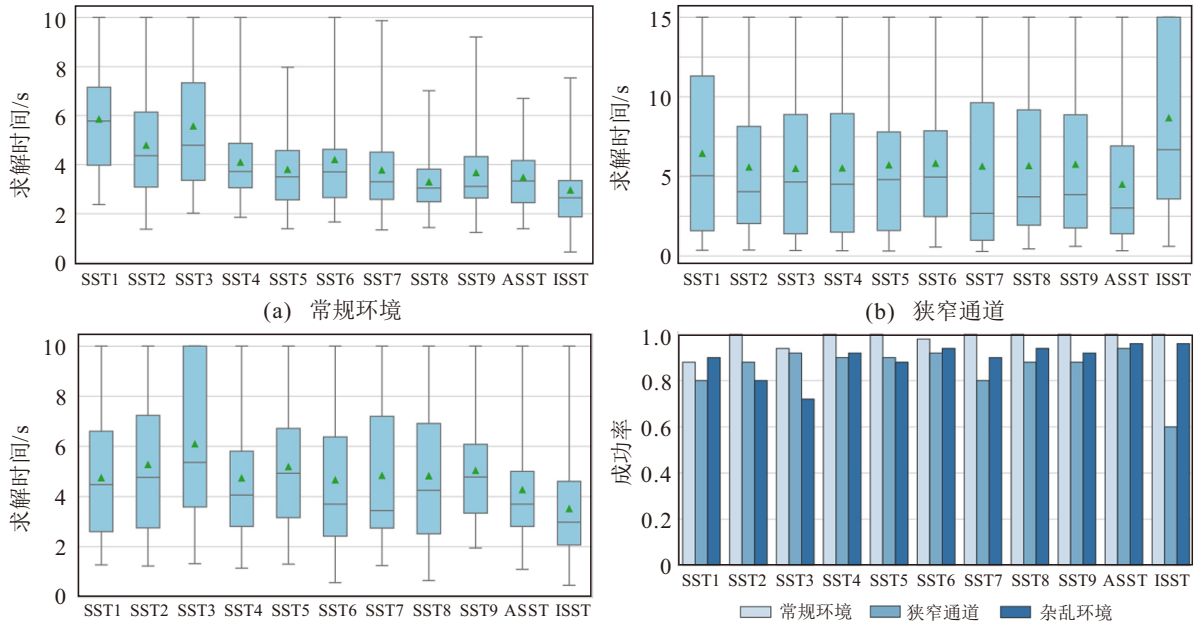
为了验证 ASST 在不同环境中的适应能力, 本节在 3 种不同类型环境下, 对 ASST 进行仿真验证, 以证明 ASST 无需精确调参即可在多种环境下表现良好. 该仿真采用双积分器系统, 分别在常规环境、狭窄通道和杂乱环境中进行仿真验证, 3 种环境地图设置如下: 常规环境地图尺寸为 100 m×100 m, 由 4 个宽度为 5 m, 长度为 30 m ～ 70 m 不等的矩形障碍物组成; 狭窄通道地图尺寸为 100 m×100 m, 通道长度为 30 m, 通道间距为 2 m; 杂乱环境地图尺寸为 100 m×100m, 包含 60 个随机生成的边长为 5 m 的正方形障碍物, ASST 算法在 3 种环境中的仿真结果如图 4 所示.



(a) 常规环境 (b) 狭窄通道 (c) 杂乱环境

图4 ASST 算法在不同环境中的仿真结果

为了直观展示 ASST 算法在规划性能上的提升程度, 本节在与 SST 算法做对比的基础上, 进一步与以提高规划效率为改进目标的 ISST 算法[23] 进行对比, 综合对比 3 种算法在不同环境下的性能表现. 本节为常规环境、狭窄通道和杂乱环境分别设置了 100 个不同的实例, 详细对比了 SST 算法的 9 组参数、ASST 算法和 ISST 算法在各种环境下的求解时间和成功率. SST 算法和 ASST 算法的仿真参数与表 1 中的双积分器一致, ISST 的δ_s取值为 2, 仿真对比如图 5 所示.



(a) 常规环境 (b) 狭窄通道

(c) 杂乱环境 (d) 不同环境下的成功率

图5 不同环境下的性能指标对比

从仿真数据中可以看出, 在常规环境中, SST8 在各组 SST 参数中表现最优, 其求解时间均值为 3.29 s, 中位数为 3.04 s; 相比之下, ISST 算法求解时间均值为 2.95 s, 中位数为 2.65 s; ASST 算法的求解时间均值为 3.47 s, 中位数为 3.33 s. 3 种算法中 ISST 的求解时间最短, 而 ASST 在求解时间上与表现最优的一组 SST 相近. 在狭窄通道中, SST3 在各组 SST 参数中表现最优, 其求解时间均值为 5.49 s, 中位数为 4.65 s; 而 ISST 算法求解时间均值为 8.67 s, 中位数为 6.68 s; ASST 算法的求解时间均值为 4.49 s, 中位数为 3.03 s, 在 3 种算法中表现最好. 在杂乱环境中, SST6 在各组 SST 参数中表现最优, 其求解时间均值为 4.66 s, 中位数为 3.69 s; 相比之下, ISST 算法求解时间均值为 3.51 s, 中位数为 2.98 s; 而 ASST 算法的求解时间均值为 4.26 s, 中位数为 3.69 s, 优于表现最好的 SST6, 但略低于 ISST 算法. 在规划成功率上, ISST 算法在常规环境、狭窄通道和杂乱环境下的规划成功率分别是 1, 0.6, 0.96; 而 ASST 算法的规划成功率分别是 1, 0.94, 0.96, 在 3 种环境中均具有最高的成功率. 从仿真结果可以看出, 在常规环境和杂乱环境中, ISST 算法的求解时间最短, 在 3 种算法中拥有最好的性能表现. 然而在狭窄通道中, ISST 算法的性能表现急剧下降, 在求解时间和求解成功率上均远低于 SST 算法和 ASST 算法. 而 ASST 算法在 3 种环境中均具有最高的求解成功率, 且在 3 种环境中均有较快的求解速度, 验证了 ASST 算法能够有效提高对复杂困难环境的求解能力, 对多种不同环境均具有良好的适应性.

3.3　渐近最优求解过程的性能指标对比

为了验证算法在迭代过程中能够逐渐收敛到近最优解, 本部分针对双积分器系统进行仿真, 以观察求解代价和节点数量的变化趋势. 仿真地图设置为杂乱环境, 尺寸为 150 m×100 m, 包含随机生成的 50 个边长小于 10 m 的矩形障碍物. 在相同环境条件下, 对双积分器系统进行了 20 次仿真, 每组仿真持续时间为 50 s, 并记录各时间点的求解代价 (选取路径距离作为求解代价) 和节点数量. 为了客观评估 ASST 算法的收敛速度和空间占用情况, 仿真选取了两组表现较优的 SST 参数作为对比参照. SST1、SST2、ASST 的参数δ_{BN}分别设置为 2、4、[2, 8]; 参数δ_s分别设置为 0.2、1、[0.2, 2]. 仿真结果如图 6 所示.

由仿真结果可见, 随着求解时间增加, ASST 的平均代价降低最为显著. 初次求解时, ASST 的平均代价为 208.82, 在最终时刻降至 191.20, 平均代价降低了 17.62; 相比之下, SST1 和 SST2 的平均代价分别降低了 5.83 和 11.38, 均低于 ASST 算法. 在内存占用方面, ASST 初次求解的平均节点数为 748, 最终时刻节点数为 2 518; 而 SST1 和 SST2 的初次求解平均节点数分别为 1 297 和 1 203, 最终时刻节点数为 4 813 和 4 154, 均高于 ASST 算法. 以上结果表明, 随着求解时间增加, ASST 能提高求解质量, 并且与 SST 算法相比有着更好的收敛性和更低的内存占用, 验证了 ASST 算法的有效性和优越性.

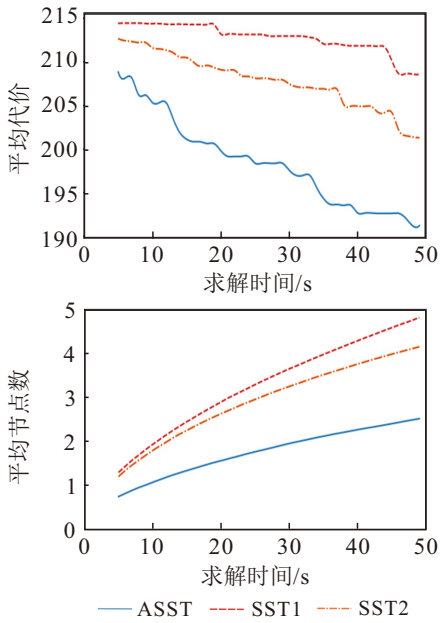


图6　平均代价和平均节点数变化曲线

4　结　论

本文提出了一种基于自适应参数的改进 SST 运动规划算法, 避免了繁琐的调参过程, 降低了算法对参数的依赖性, 提高了算法在复杂多变环境中的适用性, 提高了规划求解成功率. 然而, 在真实世界中, 系统接收到的环境信息具有不确定性, 如何针对未知的环境信息对规划的轨迹进行快速修正有待进一步研究, 是我们未来研究的方向之一.

参考文献 (References)

[1] Kavraki L E, Svestka P, Latombe J C, et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces[J]. IEEE Transactions on Robotics and Automation, 1996, 12(4): 566-580.

[2] Lavalle S M. Rapidly-exploring random trees: A new tool for path planning[R]. Ames: Iowa State University, 1998.

[3] 许万, 杨晔, 余磊涛, 等. 一种基于改进 RRT*的全局路径规划算法[J]. 控制与决策, 2022, 37(4): 829-838.
(Xu W, Yang Y, Yu L T, et al. A global path planning algorithm based on improved RRT[J]. Control and Decision, 2022, 37(4): 829-838.)

[4] 栾添添, 王皓, 孙明晓, 等. 基于动态变采样区域

RRT 的无人车路径规划[J]. 控制与决策, 2023, 38(6): 1721-1729.
(Luan T T, Wang H, Sun M X, et al. Path planning of unmanned vehicle based on dynamic variable sampling area RRT[J]. Control and Decision, 2023, 38(6): 1721-1729.)

[5] 程谦, 高嵩, 曹凯, 等. 基于 PRM 优化算法的移动机器人路径规划[J]. 计算机应用与软件, 2020, 37(12): 254-259.
(Cheng Q, Gao S, Cao K, et al. Path planning of mobile robot based on prm optimization algorithm[J]. Computer Applications and Software, 2020, 37(12): 254-259.)

[6] 钟华庚, 罗高生, 王芳, 等. 一种 Halton 序列的 HDRRT 移动机器人融合规划算法[J]. 控制与决策, 2023, 38(6): 1551-1559.
(Zhong H G, Luo G S, Wang F, et al. A Halton sequence fusion planning algorithm for HDRRT mobile robots[J]. Control and Decision, 2023, 38(6): 1551-1559.)

[7] 张腾龙, 李擎. 基于 B-RRT*FND 算法的移动机器人路径规划[J]. 控制与决策, 2023, 38(11): 3121-3127.
(Zhang T L, Li Q. Path planning of AGV based on B-RRT*FND algorithm[J]. Control and Decision, 2023, 38(11): 3121-3127.)

[8] Li L J, Miao Y L, Qureshi A H, et al. MPC-MPNet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints[J]. IEEE Robotics and Automation Letters, 2021, 6(3): 4496-4503.

[9] Donald B, Xavier P, Canny J, et al. Kinodynamic motion planning[J]. Journal of the ACM, 1993, 40(5): 1048-1066.

[10] Karaman S, Frazzoli E. Optimal kinodynamic motion planning using incremental sampling-based methods[C]. The 49th IEEE Conference on Decision and Control. Atlanta, 2010: 7681-7687.

[11] Zhang J. Kinodynamic motion planning for robotics: A review[C]. The 5th International Conference on Robotics and Automation Sciences. Wuhan, 2021: 75-83.

[12] LaValle S M. Planning algorithms[M]. New York: Cambridge University Press, 2006.

[13] Kacewicz B. Complexity of nonlinear two-point boundary-value problems[J]. Journal of Complexity, 2002, 18(3): 702-738.

[14] Li Y B, Littlefield Z, Bekris K E. Asymptotically optimal sampling-based kinodynamic planning[J]. The International Journal of Robotics Research, 2016, 35(5): 528-564.

[15] Shome R, Kavraki L E. Asymptotically optimal kinodynamic planning using bundles of edges[C]. 2021 IEEE International Conference on Robotics and Automation. Xi'an, 2021: 9988-9994.

[16] Hauser K, Zhou Y L. Asymptotically optimal planning by feasible kinodynamic planning in a state-cost space[J]. IEEE Transactions on Robotics, 2016, 32(6): 1431-1443.

[17] Nayak S, Otte M W. Bidirectional sampling-based motion planning without two-point boundary value solution[J]. IEEE Transactions on Robotics, 2022, 38(6): 3636-3654.

[18] Kleinbort M, Solovey K, Littlefield Z, et al. Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation[J]. IEEE Robotics and Automation Letters, 2023, 8(2): 1149-1150.

[19] Papadopoulos G, Kurniawati H, Patrikalakis N M. Analysis of asymptotically optimal sampling-based motion planning algorithms for lipschitz continuous dynamical systems[J/OL]]. 2014, arXiv: 1405.2872.

[20] 余卓平, 李奕姗, 熊璐. 无人车运动规划算法综述[J]. 同济大学学报: 自然科学版, 2017, 45(8): 1150-1159.
(Yu Z P, Li Y S, Xiong L. A review of the motion planning problem of autonomous vehicle[J]. Journal of Tongji University: Natural Science, 2017, 45(8): 1150-1159.)

[21] Littlefield Z, Bekris K E. Informed asymptotically near-optimal planning for field robots with dynamics[C]. Field and Service Robotics. Cham: Springer, 2018: 449-463.

[22] Kalisiak M, van de Panne M. RRT-blossom: RRT with a local flood-fill behavior[C]. Proceedings 2006 IEEE International Conference on Robotics and Automation. Orlando, 2006: 1237-1242.

[23] Hu F Y, Cui B, Mu S B, et al. A forward propagation motion planning algorithm based on generative model[C]. 2023 IEEE International Conference on Unmanned Systems. Hefei, 2023: 1201-1206.

[24] Arteaga R, Antonio E, Becerra I, et al. On the efficiency of the SST planner to find time optimal trajectories among obstacles with a DDR under second order dynamics[J]. IEEE Robotics and Automation Letters, 2022, 7(2): 674-681.

[25] Littlefield Z, Bekris K E. Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions[C]. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. Madrid. 2018: 1-9.

作者简介

崔冰 (1990−), 男, 副教授, 博士生导师, 主要研究方向为多智能体轨迹规划与协同控制、航天器协同控制, E-mail: bing.cui@bit.edu.cn;

李广 (2002−), 男, 硕士生, 主要研究方向为多智能体轨迹规划, E-mail: 3120230845@bit.edu.cn;

胡飞扬 (1999−), 男, 硕士生, 主要研究方向为机器人运动规划, E-mail: 360452176@qq.com;

高寒 (1991−), 男, 副研究员,硕士生导师, 主要研究方向为航天器姿态控制、故障诊断及容错控制, E-mail: gaohbit@ bit.edu.cn;

夏元清 (1971−), 男, 教授, 博士生导师, 主要研究方向为云控制技术、天空地海一体化网络跨域协同控制, E-mail: xia_yuanqing@bit.edu.cn.