文章编号: 1001-0920 (2015) 10-1751-08 DOI: 10.13195/j.kzyjc.2014.1282

一种新的实时智能汽车轨迹规划方法

付骁鑫[a], 江永亨[a], 黄德先[a], 黄开胜[b], 王京春[a], 陆 耿[a]

(清华大学 a. 自动化系, b. 汽车工程系, 北京 100084)

摘 要: 针对智能汽车的驾驶决策和轨迹规划问题, 将轨迹表示为轨迹曲线和加速度变化两部分, 以优化轨迹的行驶效率、安全性、舒适性和经济性为目标建立非线性规划模型. 基于序优化思想, 提出混合智能优化算法 OODE, 分内、外两层分别优化加速度变化和轨迹曲线, 通过“粗糙”评价轨迹曲线实现轨迹曲线的快速择优. 仿真结果表明, 所提出的方法能够处理包含多动态障碍物的复杂交通场景, 且具备实时应用能力, 模型的精度和求解速度均优于传统方法.

关键词: 轨迹规划; 智能汽车; 序优化; 粗糙评价

中图分类号: TP273 **文献标志码:** A

A novel real-time trajectory planning algorithm for intelligent vehicles

FU Xiao-xin[a], *JIANG Yong-heng*[a], *HUANG De-xian*[a], *HUANG Kai-sheng*[b], *WANG Jing-chun*[a], *LU Geng*[a]

(a. Department of Automation, b. Department of Automotive Engineering, Tsinghua University, Beijing 100084, China. Correspondent: JIANG Yong-heng, E-mail: jiangyh@tsinghua.edu.cn)

Abstract: With the trajectory modeled in two parts: trajectory curve and acceleration profile, the problems of decision-making and trajectory planning for intelligent vehicles are formulated as a non-linear programming(NLP) model to optimize the efficiency, safety, comfort and economy of trajectory. To solve this model, a hybrid intelligent optimization algorithm OODE is developed. With a two-layer framework applied, OODE optimizes the acceleration profile and trajectory curve in the inner and outer layers, respectively. By “roughly” evaluating the candidate trajectory curves, the optimal curve is determined very efficiently. The simulation results show that, the proposed method is capable of handling complicated traffic scenarios with multiple dynamic obstacles, and also can meet the demands of real-time applications. Compared with traditional methods, the model accuracy of the proposed method is higher, and the planning speed is obviously faster.

Keywords: trajectory planning; intelligent vehicle; ordinal optimization; rough evaluation

0 引 言

伴随着信息和传感器技术的进步, 智能汽车的出现改善了交通运输效率, 提升了驾驶的舒适度和安全性, 同时将人类从繁杂的驾驶任务中解放出来, 吸引了全世界研究人员的关注. 轨迹规划指的是生成未来一段时间内的车辆行驶轨迹, 是实现智能汽车的一项关键性技术, 在近年来有丰富的研究成果.

一些研究者关注一维轨迹规划, 在确定的轨迹曲线上, 通过分析周围障碍物的相对运动状态, 结合运动模型 (如刹车模型[1]) 计算碰撞时间、制动距离等统计指标, 进而规划行驶轨迹.

一些研究者将智能汽车看作一类结构特殊的机器人, 将原本应用在机器人领域的路径规划算法移植过来用于生成车辆轨迹, 如势场法[2-3]. 鉴于势场法生成的轨迹在平滑性上的不足, 研究人员进一步提出了橡皮筋算法 (elastic band algorithms)[4]. 这两类方法虽然实现简单, 且具备实时工作的能力, 但难以处理动态变化的驾驶环境和车辆的运动学约束, 同时所求解易陷入局部最优. 搜索算法 (如 A*[5], D*[6], RRT[7]) 是另一类广泛应用的规划方法. 这类方法通过构造特殊的构形空间[8]和引入重规划机制[9], 可以处理运动学约束和动态环境, 但求解效率低, 消耗存储量大.

此外, 研究者还提出了基于参数优化的轨迹规划方法. 例如: 将轨迹中车辆的位置或者加速度表示为

收稿日期: 2014-08-20; **修回日期:** 2015-01-20.

基金项目: 国家自然科学基金项目(61273039); 清华大学自主科研计划项目(2011THZ0).

作者简介: 付骁鑫(1989–), 男, 博士生, 从事智能汽车、轨迹规划的研究; 江永亨(1974–), 男, 副教授, 从事智能汽车、复杂系统分析建模与优化理论方法等研究.

关于时间的函数[10-12]或者不同阶次的螺线[13-14], 然后通过优化函数参数来获得最优轨迹; 或者, 首先生成多条不同形状的、不同速度变化方式的候选轨迹, 然后对它们作出评价和比较来找出最优者[6,15]. 上述方法的特点是通过简化轨迹模型来减少待求变量的个数, 使轨迹规划的难度降低, 从而保证求解速度, 但其求解质量会受到轨迹描述方式的限制.

不同于已有方法, 本文关注的是现实世界的繁忙公路环境中智能汽车的实时轨迹规划问题, 其交通场景具有以下特点:

1) 动态障碍物(目前只考虑车辆)较多;

2) 行驶环境时刻处于快速的动态变化中;

3) 行驶道路的几何分布高度结构化;

4) 行驶环境中的所有交通参与对象的运动均遵循严格规则.

上述特点使得智能汽车的运动环境趋于高度复杂化, 评价轨迹的计算复杂度很高, 同时要求规划结果随环境的动态变化不断更新, 对规划速度提出了实时性要求. 注意到, 大多数时间内, 全部交通参与对象均在固定车道内沿道路中心线运动, 且任意时刻本车(即执行轨迹的车辆)要么处于换道行驶状态要么处于车道保持状态, 其驾驶决策是简单直接的. 因此, 本文选择将智能汽车的驾驶决策和轨迹规划放在同一个问题中解决, 即把换道行驶和车道保持对应的多条轨迹一起送入规划器中. 通过对它们作比较和筛选决定是否进行换道, 同时生成最优轨迹. 这样做, 一方面确保了决策是依据对应轨迹的量化评价制定的, 保证了轨迹求解质量; 另一方面, 同时完成决策和规划, 效率更高. 此外, 从优化的角度看, 轨迹规划可看作是在可行轨迹构成的解空间内寻找最优解, 而驾驶决策其实是为了缩小解空间的搜索范围以提高轨迹规划效率, 并不一定要在规划前完成.

根据以上讨论, 本文将轨迹表示为轨迹曲线和加速度变化两部分, 针对智能汽车的驾驶决策和轨迹规划问题, 以最优化轨迹的综合性能(行驶效率、安全性、舒适性和经济性)为目标建立统一的非线性规划(NLP)模型, 并基于序优化思想提出混合智能优化算法OODE进行求解.

1 优化模型

1.1 问题描述

轨迹规划问题可以定义为: 已知车辆的起始运动状态, 考虑驾驶目的和驾驶环境的变化, 生成一条综合性能最优的行驶轨迹. 规划结果包括二维平面上的行驶路线和行驶速度沿路线的变化过程. 轨迹规划要考虑的约束条件包括车辆的不完全运动学约束和运动状态量的物理约束. 规划结果必须保证轨迹曲线光滑且曲率连续, 车辆加速度和速度都随时间连续变化, 并满足各自的边界条件. 轨迹的性能评价需兼顾行驶效率和驾驶的安全性、舒适性、经济性, 特别是安全性, 车辆行驶过程应避免任何碰撞, 并与周围障碍物保持一定的安全间距. 本文从优化角度出发, 将轨迹规划问题建模为NLP模型, 以期设计合理的优化策略平衡求解质量和求解速度, 保证求解效率.

1.2 轨迹表示

分轨迹曲线和加速度变化两部分对车辆轨迹进行建模. 鉴于多项式[8]在求导和计算上的便利性, 本文将轨迹曲线表示为轨迹上任一点的纵坐标y关于横坐标x的τ次多项式函数, 表示为

$$\begin{aligned} y = g(x,\boldsymbol{b}) = b_0 + b_1 x + b_2 x^2 + \cdots + b_\tau x^\tau;\\ x_0 \leqslant x \leqslant x_f, \boldsymbol{b} \in B^K. \end{aligned} \tag{1}$$

其中: $\boldsymbol{b} = [b_0 \quad b_1 \quad \cdots \quad b_\tau]^{\mathrm{T}}$是轨迹曲线参数向量, B^K是包含$\boldsymbol{b}$可能取值的$K(K=\tau+1)$维空间, x_0和x_f分别是轨迹起始点和结束点的横坐标值.

轨迹的加速度变化被表示为车辆加速度的离散序列. 将轨迹曲线等长度地划分为N个轨迹片段, 假定在各轨迹片段上本车(执行轨迹的车辆)以常量加速度运动. 用向量$\boldsymbol{a}$表示各片段上的加速度$a_1, a_2, \cdots, a_N$, 即$\boldsymbol{a} = [a_1 \quad a_2 \quad \cdots \quad a_N]^{\mathrm{T}}, \boldsymbol{a} \in R^N$, 则$\boldsymbol{a}$决定了轨迹上任一点的速度.

轨迹片段的长度Δs_f计算如下:

$$\Delta s_f = \frac{1}{N}\int_{x_0}^{x_f} \sqrt{1+(g'(x,\boldsymbol{b}))^2}\mathrm{d}x. \tag{2}$$

应用Simpson积分公式

$$\int_a^b f(x)\mathrm{d}x \approx \frac{(b-a)\cdot[f(a)+4f((a+b)/2)+f(b)]}{6},$$

式(2)可以简化计算为

$$\begin{aligned} \Delta s_f \approx\ & \frac{1}{N}\cdot\frac{\Delta x}{6}\sum_{m=1}^{M}\{p(x_0+(m-1)x)+\\ & 4p(x_0+(m-1/2)\Delta x)+p(x_0+m\Delta x)\}. \end{aligned} \tag{3}$$

其中

$$p(x) = \sqrt{1+(g'(x,\boldsymbol{b}))^2},$$

M为应用Simpson公式将轨迹曲线按横坐标均匀划分的间隔数, $\Delta x = (x_f - x_0)/M$为横坐标间隔的长度. 各轨迹片段结束点的横坐标x_n之间的关系为

$$\int_{x_{n-1}}^{x_n} \sqrt{1+(g'(x,\boldsymbol{b}))^2}\mathrm{d}x = \Delta s_f. \tag{4}$$

已知$(\boldsymbol{b},\boldsymbol{a})$和$x_{n-1}$, 利用式(3)和(4)可递推计算$x_n$. 进一步, 设轨迹起始点的速度和加速度为$v_0$和$a_0$, 可计算在第$n$个轨迹片段结束点本车的其他运

动状态量(纵坐标 y_n, 车身方向角 θ_n, 速度 v_n, 行驶时间 t_n)

$$\begin{cases} y_n = g(x_n, \boldsymbol{b}); \\ \theta_n = \arctan g'(x_n, \boldsymbol{b}); \\ v_n = \sqrt{v_{n-1}^2 + 2a_n \cdot \Delta s_f}; \\ t_n = t_{n-1} + 2\Delta s_f/(v_n + v_{n-1}),\ t_0 = 0. \end{cases} \tag{5}$$

由式(5)可知, 利用 $(\boldsymbol{b}, \boldsymbol{a})$ 可以计算本车在各个轨迹片段结束点的运动状态 $(x_n, y_n, \theta_n, a_n, v_n, t_n)$. 因此, 利用 $(\boldsymbol{b}, \boldsymbol{a})$ 可唯一确定一条轨迹, 优化模型中将 $(\boldsymbol{b}, \boldsymbol{a})$ 定义为优化变量.

1.3　数学模型

1.3.1　约束条件

在 1.2 节引入的轨迹表示方法已确保了轨迹曲线光滑和曲率连续, 优化模型的约束条件应保证 $\boldsymbol{a}$ 中各段加速度满足边界约束 $C_a^L \leqslant a_n \leqslant C_a^U$ (C_a^U 和 C_a^L 分别是加速度的上界和下界), 记为

$$h_u(\boldsymbol{a}) \leqslant 0,\ u = 1, 2, \cdots, 2N. \tag{6}$$

其中

$$h_{2n-1}(\boldsymbol{a}) = a_n - C_a^U;$$
$$h_{2n}(\boldsymbol{a}) = C_a^L - a_n,\ n = 1, 2, \cdots, N.$$

1.3.2　目标函数

优化模型的目标函数 $J(\boldsymbol{b}, \boldsymbol{a})$ 需兼顾轨迹的行驶效率、安全性、经济性和舒适性, 描述轨迹的综合性能, 因此设计轨迹的总时间代价 C_{Time}, 加速度平方代价 C_{Acce}, 加速度增量平方代价 C_{dAcce}, 不合法速度代价 C_{Speed} 和碰撞安全代价 C_{Coll} 等 5 项评价指标, 它们的物理意义、计算方法以及对轨迹的影响如表 1 所示.

表 1　轨迹的评价指标

指标	物理意义	计算方法	对轨迹的影响
C_{Time}	总行驶时间	t_n	行驶效率
C_{Acce}	加速度的平方和	$\sum_{n=1}^{N} a_n^2 \cdot \Delta s_f$	舒适性、安全性
C_{dAcce}	加速度增量的平方和	$\sum_{n=1}^{N} \Delta a_n^2 \cdot \Delta s_f$	舒适性、经济性
C_{Speed}	速度安全风险的和	$\sum_{n=1}^{N} \eta_n \cdot \Delta s_f$	安全性
C_{Coll}	碰撞安全风险的和	$\sum_{n=1}^{N} e_n \cdot \Delta s_f$	安全性

表 1 中, $\Delta a_n = a_n - a_{n-1}$, C_{Time}、C_{Acce} 和 C_{dAcce} 可以根据本车的运动状态直接计算. C_{Speed} 表示轨迹的不合法速度引入的安全风险, 第 n 个轨迹片段的不合法速度风险 η_n 计算为

$$\eta_n = \begin{cases} 1,\ v_n > v_{\text{spd}} \text{ or } v_n < 0; \\ 0,\ \text{else}. \end{cases} \tag{7}$$

其中 v_{spd} 表示允许的最大行驶速度. C_{Coll} 表示碰撞引入的安全风险, 第 n 个轨迹片段的碰撞风险 e_n 计算为

$$e_n = e_{\alpha,n} + e_{\beta,n} + \cdots,$$

其中 $e_{\alpha,n}, e_{\beta,n}, \cdots$ 分别表示 t_n 时刻本车与障碍物 α, $\beta, \cdots$ 间的碰撞安全风险.

下面以 $e_{\alpha,n}$ 为例说明碰撞安全风险的计算. 设 t_n 时刻障碍物 α 与本车EV车身中心的距离为 $d_{\alpha,n}$ ($d_{\alpha,n} > 0$), α 和 EV 的速度方向与两者连接线的夹角分别为 $\xi_{\alpha,n}$ 和 $\delta_{\alpha,n}$, α 与 EV 在连接线方向上的相对速度为 $\text{rv}_{\alpha,n}$. $\text{rv}_{\alpha,n} > 0$ 表示 α 与 EV 相互远离, $\text{rv}_{\alpha,n} < 0$ 表示 α 与 EV 相互接近. 上述几何变量的关系如图 1 所示, 上方的矩形表示障碍物 α, 下方的矩形表示本车 EV.

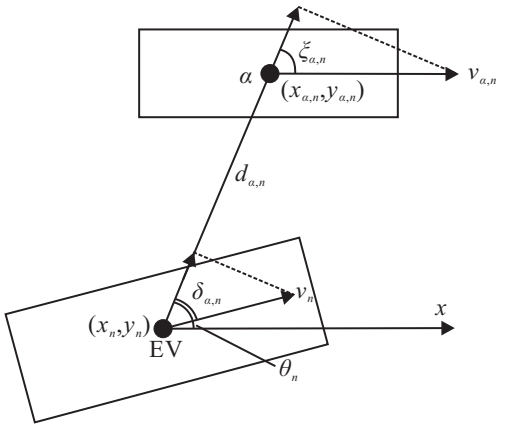


图 1　碰撞安全风险的计算

已知 t_n 时刻 α 的位置 $(x_{\alpha,n}, y_{\alpha,n})$ 和速度 $v_{\alpha,n}$, 根据下式计算 $e_{\alpha,n}$:

$$\begin{cases} d_{\alpha,n} = \sqrt{(x_{\alpha,n} - x_n)^2 + (y_{\alpha,n} - y_n)^2}, \\ \cos\xi_{\alpha,n} = (x_{\alpha,n} - x_n)/d_{\alpha,n}, \\ \sin\xi_{\alpha,n} = (y_{\alpha,n} - y_n)/d_{\alpha,n}, \\ \cos\delta_{\alpha,n} = \cos\theta_n \cdot \cos\xi_{\alpha,n} + \sin\theta_n \cdot \sin\xi_{\alpha,n}, \\ \text{rv}_{\alpha,n} = v_{\alpha,n} \cdot \cos\xi_{\alpha,n} - v_n \cdot \cos\delta_{\alpha,n}, \\ e_{\alpha,n} = \exp(-0.1 \cdot rv_{\alpha,n})/d_{\alpha,n}. \end{cases} \tag{8}$$

利用表 1 中的各项评价指标, 优化模型的目标函数 $J(\boldsymbol{b}, \boldsymbol{a})$ 定义为它们的加权和, 即

$$\begin{aligned} J(\boldsymbol{b}, \boldsymbol{a}) = {} & \\ & w_{\text{Time}} \cdot C_{\text{Time}} + w_{\text{Acce}} \cdot C_{\text{Acce}} + \\ & w_{\text{dAcce}} \cdot C_{\text{dAcce}} + \\ & w_{\text{Speed}} \cdot C_{\text{Speed}} + w_{\text{Coll}} \cdot C_{\text{Coll}}, \end{aligned} \tag{9}$$

其中 w_{Time}, w_{Acce}, w_{dAcce}, w_{Speed}, w_{Coll} 分别是评价指标 C_{Time}, C_{Acce}, C_{dAcce}, C_{Speed}, C_{Coll} 的权重.

综合前述内容, 轨迹规划问题建模为如下的 NLP 模型:

$$\Pi:\ \min_{\boldsymbol{b},\boldsymbol{a}} J(\boldsymbol{b},\boldsymbol{a}),\ \boldsymbol{b}\in B^K,\boldsymbol{a}\in R^N;$$
$$\text{s.t. } h_u(\boldsymbol{a})\leqslant 0,\ u=1,2,\cdots,2N. \tag{10}$$

问题Π的特点, 同时也是求解难点, 概括如下:

1) 解空间大. 优化变量中向量$\boldsymbol{a}$的长度等于轨迹片段的划分段数, 轨迹解空间的大小随着轨迹划分段数的增加指数级增大.

2) 评价解的计算量大. 轨迹解评价涉及复杂非线性评价指标的计算(如$C_{\rm Coll}$), 因此评价耗时长.

3) 优化求解的速度要求高. 轨迹规划需要实时进行, 与车辆行驶速度匹配, 因此其求解速度必须满足实时性要求.

2 优化算法

2.1 算法框架

针对问题Π的特点, 本文提出一种将序优化思想与传统智能优化算法结合的混合优化算法进行求解. 首先从物理概念出发, 将原问题重写为两层结构, 外层优化轨迹曲线, 内层在固定的轨迹曲线上优化加速度序列.

$$\Pi':\ \min_{\boldsymbol{b}}(\min_{\boldsymbol{a}} J(\boldsymbol{b},\boldsymbol{a})),\ \boldsymbol{b}\in B^K,\boldsymbol{a}\in R^N;$$
$$\text{s.t. } h_u(\boldsymbol{a})\leqslant 0,\ u=1,2,\cdots,2N. \tag{11}$$

进一步, 定义$\boldsymbol{a}$为内层优化变量, $\boldsymbol{b}$为外层优化变量, 将问题Π′分解为内层问题Π″和外层问题Π‴

$$\Pi'':\ J_b(\boldsymbol{b})=\min_{\boldsymbol{a}} J(\boldsymbol{b},\boldsymbol{a}),\ \boldsymbol{a}\in R^N;$$
$$\text{s.t. } h_u(\boldsymbol{a})\leqslant 0,\ u=1,2,\cdots,2N. \tag{12}$$
$$\Pi''':\ \min_{\boldsymbol{b}} J_b(\boldsymbol{b}),\ \boldsymbol{b}\in B^K. \tag{13}$$

由式(11)～(13)可知, 问题Π′的求解是通过在外层问题Π‴的求解中嵌套内层问题Π″的求解完成的. 外层问题Π‴的评价值$J_b(\boldsymbol{b})$就是内层问题Π″的目标函数最优值$\min_{\boldsymbol{a}} J(\boldsymbol{b},\boldsymbol{a})$, 外层优化对外层变量$\boldsymbol{b}$的评价过程就是内层问题Π″的求解过程.

序优化(OO)认为, "1) 序比值更容易(确定$A>B$是否成立比求出$A-B$的值更容易); 2) 软化求解目标能使问题简化(确定地求出最优解不如以高概率找到足够好的解)"[16]. 也就是说, 通过"粗糙"但快速地进行解的评价和择优(即只关心解的优劣顺序), 同时将优化目标软化为求取满意解, 能提高求解效率.

考察问题Π′, 在外层问题Π‴的求解中应用序优化思想, 对内层问题Π″对应的优化模型做粗糙化, 通过"粗糙"求解问题Π″, 即求取满意解(足够好的解), 快速计算关于外层优化变量$\boldsymbol{b}$的粗糙评价$\hat{J}_b(\boldsymbol{b})$. 粗糙化可从模型和求解两个角度进行: 模型上, 可选择增大模型粒度从而降低优化问题的规模, 或简化约束条件从而降低求解难度等; 求解上, 可选择修改解的迭代更新机制(如加强局部搜索以加快收敛速度), 或设置松弛的优化终止规则(如在完全收敛前退出优化)等. 粗糙评价模型如下:

$$\Pi_c'':\ \hat{J}_b(\boldsymbol{b})=\min_{\boldsymbol{a}} J(\boldsymbol{b},\boldsymbol{a}),\ \boldsymbol{a}\in R^{N_c};$$
$$\text{s.t. } h_{u,c}(\boldsymbol{a})\leqslant 0,\ u=1,2,\cdots,U_c. \tag{14}$$

其中: N_c为增大模型粒度后$\boldsymbol{a}$中变量的个数, $h_{u,c}(\boldsymbol{a})\leqslant 0$为简化后的约束. 求解$\Pi_c''$后, 将$\hat{J}_b(\boldsymbol{b})$看作对由$\boldsymbol{b}$决定的轨迹曲线的评价, 通过比较不同轨迹曲线的$\hat{J}_b(\boldsymbol{b})$可以十分高效地确定最优曲线. 然后基于最优曲线, 再次严格地求解问题Π″, 得到原问题Π的最终解. 这里为了与粗糙评价模型区分开, 将严格求解的优化模型Π″称为精确评价模型, 对应得到的轨迹曲线评价称为精确评价. 据此, 本文以差分进化算法(DE)[17]作为求解内层问题Π″的子算法, 提出混合智能优化算法OODE, 其算法框架如图2所示.

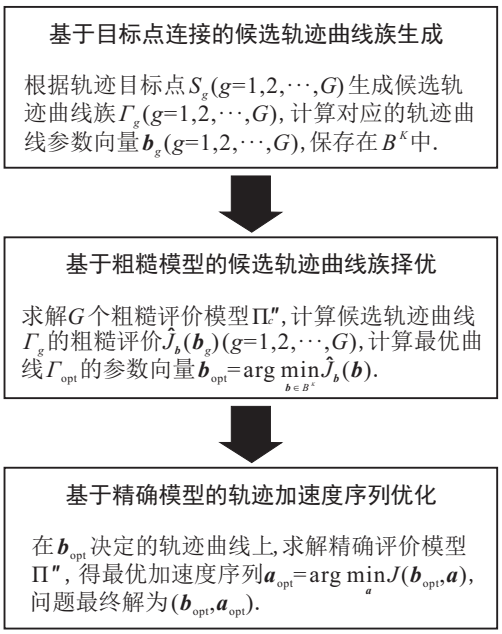


图 2 混合智能优化算法 OODE 的算法框架

OODE算法的求解分为3个阶段:

1) 基于目标点连接的候选轨迹曲线族生成.

选取G个轨迹目标点$S_g(g=1,2,\cdots,G)$, 生成候选轨迹曲线族$\Gamma_g\ (g=1,2,\cdots,G)$, 分别连接轨迹起始点$T$和目标点$S_g\ (g=1,2,\cdots,G)$, 计算$\Gamma_g$的轨迹曲线参数向量$\boldsymbol{b}_g\ (g=1,2,\cdots,G)$, 并保存在$B^K$中.

2) 基于粗糙模型的候选轨迹曲线族择优.

应用DE算法求解G个粗糙评价模型

$$\Pi_c'':\min_{\boldsymbol{a}} J(\boldsymbol{b}_g,\boldsymbol{a})\ (g=1,2,\cdots,G),$$

计算目标函数最优值作为对轨迹曲线Γ_g的粗糙评价$\hat{J}_b(\boldsymbol{b}_g)$, 比较确定最优轨迹曲线$\Gamma_{\rm opt}$, 其参数向量

$$\boldsymbol{b}_{\rm opt}=\arg\min_{\boldsymbol{b}\in B^K}\hat{J}_b(\boldsymbol{b}).$$

3) 基于精确模型的轨迹加速度序列优化.

应用 DE 算法求解精确评价模型 Π'' : $\min\limits_{\boldsymbol{a}} J(\boldsymbol{b}_{\rm opt}, \boldsymbol{a})$, 得到 $\boldsymbol{a}_{\rm opt}$.

问题的最终解为 $(\boldsymbol{b}_{\rm opt}, \boldsymbol{a}_{\rm opt})$.

2.2 算法实现

2.2.1 基于目标点连接的候选轨迹曲线族生成

在第 1 阶段, OODE 算法生成连接固定起始点 T 和不同目标点 S_g $(g = 1, 2, \cdots, G)$ 的候选轨迹曲线 Γ_g $(g = 1, 2, \cdots, G)$, 计算它们的参数向量 $\boldsymbol{b}_g$ $(g = 1, 2, \cdots, G)$, 并保存在集合 B^K 中.

在已实现智能汽车的规划框架[6,18]中, 目标点的指定通常由其他更高层的规划模块(如行为规划模块)完成, 本文假定目标点已知. 不失一般性, 这里以 3 条车道内沿中心线分布的 G $(G = 2R + 1)$ 个点 $S_1, S_2, \cdots, S_G$ 为目标点, 由左相邻车道内的 R 个点、右相邻车道内的 R 个点和当前车道内的 1 个点构成. 左右车道内的目标点以等间隔 Δl 分布, 分布中心和当前车道内目标点与轨迹起始点的纵向距离均为 l, 因此可确定各目标点的位置和前进方向(用斜率表示).

本车沿所生成轨迹行驶时应确保到达指定目标点, 因此轨迹曲线在结束点的位置和斜率应与目标点一致. 设起始点的位置坐标和斜率为 (x_0, y_0) 和 k_0, 第 g 个目标点 S_g 的位置坐标和斜率为 (x_g, y_g) 和 k_g, 则轨迹曲线 Γ_g 的参数向量 $\boldsymbol{b}_g$ 应满足如下方程:

$$\begin{cases} g(x_0, \boldsymbol{b}_g) = y_0, \\ g'(x_0, \boldsymbol{b}_g) = k_0, \\ g(x_g, \boldsymbol{b}_g) = y_g, \\ g'(x_g, \boldsymbol{b}_g) = k_g. \end{cases} \tag{15}$$

易知, $g(x, \boldsymbol{b}_g)$ 至少应是三次多项式才能保证式 (15) 有解, 这里取 $\tau = 3$ (当 $\tau > 3$ 时, $\boldsymbol{b}_g$ 中将存在自由参数可用于进一步优化曲线, 但在本文中并非研究重点). 直接求解式 (15), $\boldsymbol{b}_g$ 按下式计算:

$$\begin{cases} b_3 = \dfrac{(k_g + k_0)(x_g - x_0) - 2(y_g - y_0)}{(x_g - x_0)^3}, \\ b_2 = \dfrac{k_g - k_0}{2(x_g - x_0)} - \dfrac{3}{2} b_3 (x_g + x_0), \\ b_1 = k_0 - 2b_2 x_0 - 3b_3 x_0^2, \\ b_0 = y_0 - b_1 x_0 - b_2 x_0^2 - b_3 x_0^3. \end{cases} \tag{16}$$

代入不同目标点的位置和斜率, 可计算候选轨迹曲线族的参数向量 $\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_G$, 外层优化变量 $\boldsymbol{b}$ 在离散集合 $B^K = \{\boldsymbol{b}_g,\ g = 1, 2, \cdots, G\}$ 中取值.

2.2.2 基于粗糙模型的候选轨迹曲线族择优

OODE 算法在第 2 阶段用 DE 算法求解粗糙评价模型 Π_c'', 计算对 Γ_g 的粗糙评价 $\hat{J}_b(\boldsymbol{b}_g)$, 模型的粗糙化具体采用以下两个策略:

1) 增大模型粒度. 对轨迹曲线作更粗糙的划分, 即轨迹片段的段数取为 N_c $(N_c < N)$, 使得问题的优化变量个数减小, 问题规模下降.

2) 减少迭代次数. 在问题 Π'' 的优化求解过程中, $\min\limits_{\boldsymbol{a}} J(\boldsymbol{b}_g, \boldsymbol{a})$ 随迭代次数 I 的增加而逐渐收敛. 但 I 越大, 每次迭代对 $\min\limits_{\boldsymbol{a}} J(\boldsymbol{b}_g, \boldsymbol{a})$ 的改善越不明显. 因此, 不必将问题精确求解至收敛曲线趋于平坦, 可在更早时刻退出. 不必迭代 I_0 次(充分迭代), 在 $I = I_c$ $(I_c < I_0)$ 时可结束当前优化.

根据得到的轨迹曲线 $\Gamma_1, \Gamma_2, \cdots, \Gamma_G$ 的粗糙评价 $\hat{J}_b(\boldsymbol{b}_1), \hat{J}_b(\boldsymbol{b}_2), \cdots, \hat{J}_b(\boldsymbol{b}_G)$, 最优轨迹曲线 $\Gamma_{\rm opt}$ 的参数向量 $\boldsymbol{b}_{\rm opt}$ 计算为

$$\boldsymbol{b}_{\rm opt} = \arg\min_{\boldsymbol{b} \in B^K} \hat{J}_b(\boldsymbol{b}). \tag{17}$$

2.2.3 基于精确模型的轨迹加速度序列优化

在最后阶段, DE 算法被用于求解精确评价模型 Π'', 计算 $\Gamma_{\rm opt}$ 对应的最优加速度向量

$$\boldsymbol{a}_{\rm opt} = \arg\min_{\boldsymbol{a}} J(\boldsymbol{b}_{\rm opt}, \boldsymbol{a}). \tag{18}$$

精确评价模型与粗糙评价模型的区别是优化模型采用原始模型粒度 (即轨迹曲线被划分为 N 段), 且求解算法 DE 在迭代充分 (即 $I = I_0$) 后退出. 最后, $(b_{\rm opt}, a_{\rm opt})$ 为原问题 Π 的最终解.

3 仿真结果

本节对 OODE 算法的结果和性能进行分析, 全部仿真在搭载 Intel(R) Core(TM) i5 CPU 内存 2.0 GB 笔记本电脑的 Visual Studio 2010 平台上进行. 仿真考虑了两个交通场景 (场景 1 和场景 2), 均包含 3 个动态障碍物 (车辆), 假设以常速沿车道中心线运动. 障碍物 α, β, γ 和本车 EV 的起始运动状态见表 2 和表 3. 其中: V_1 和 D_1 表示场景 1 中障碍物的行驶速度和纵向位置, V_2 和 D_2 表示场景 2 中障碍物的行驶速度和纵向位置; EV 在场景 1 和场景 2 下的起始运动状态相同, SA、SV 和 SD 分别表示起始时刻 EV 的行驶加速度、行驶速度和纵向位置. 场景 1 和场景 2 的结构相同, 仅障碍物 γ 的运动状态不同.

表 2　障碍物的起始运动状态

障碍物	行驶车道	V_1/(feet/s)	D_1/(feet)	V_2/(feet/s)	D_2/(feet)
α	A	30	30	30	30
β	B	20	80	20	80
γ	C	10	−10	30	20

表 3　本车的起始运动状态

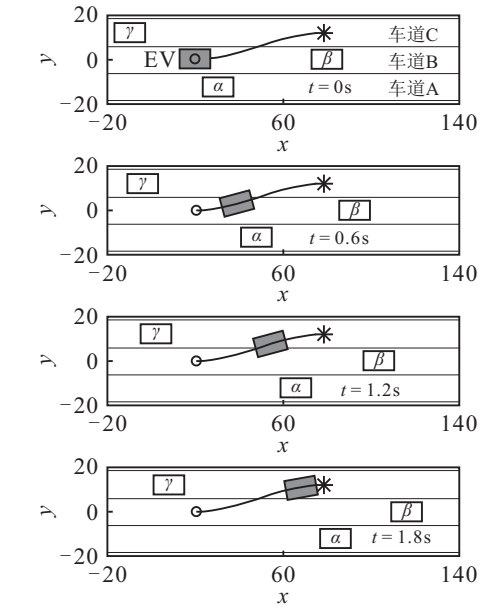
本车	行驶车道	SA/(feet/s^2)	SV/(feet/s)	SD/(feet)
EV	B	0	40	20

OODE 算法的参数取值如下: $G = 19$ 是轨迹目标点的个数; $l = 70$ feet 是轨迹目标点的分布中心与起始点的纵向距离; $\Delta l = 3$ feet 是目标点的分布间隔;

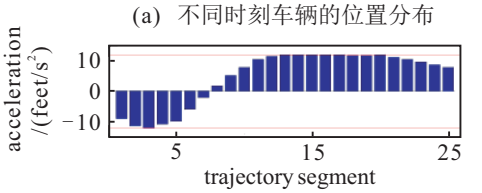
$C_a^U = 12\,\text{feet/s}^2$ 和 $C_a^L = -12\,\text{feet/s}^2$ 分别是加速度的上界和下界; $w_{\text{Time}} = 10, w_{\text{Acce}} = 3\text{e-}5, w_{\text{dAcce}} = 5\text{e-}4, w_{\text{Speed}} = 200$ 和 $w_{\text{Coll}} = 2$ 分别是评价指标 $C_{\text{Time}}, C_{\text{Acce}}, C_{\text{dAcce}}, C_{\text{Speed}}$ 和 C_{Coll} 在目标函数中的权重; $I_c = 15, N_c = 5, \text{NP} = 10_c, F_c = 0.85$ 和 $\text{CR}_c = 0.95$ 分别是用于求解 Π_c'' 的 DE 算法的迭代次数、优化变量个数、种群个体数、差分权重和交叉概率; $I_0 = 100, N = 25, \text{NP} = 50, F = 0.85$ 和 $\text{CR} = 0.95$ 分别是用于求解 Π'' 的 DE 算法的迭代次数、优化变量个数、种群个体数、差分权重和交叉概率.

3.1 轨迹规划结果

应用 OODE 算法对场景 1 和场景 2 作轨迹规划, 规划结果分别如图 3 和图 4 所示.



(a) 不同时刻车辆的位置分布

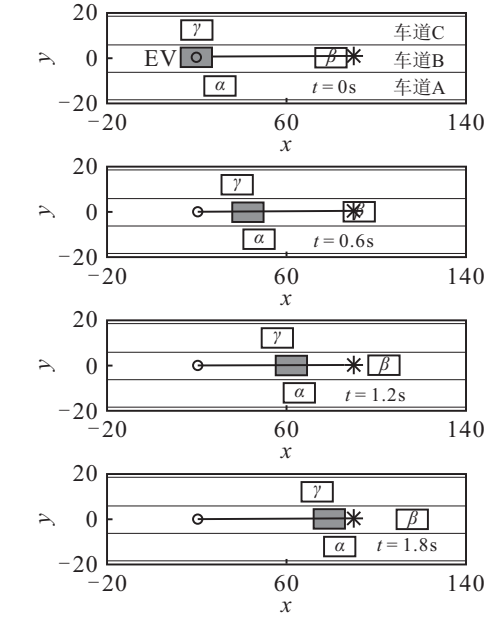


(b) 加速度在各轨迹片段的取值

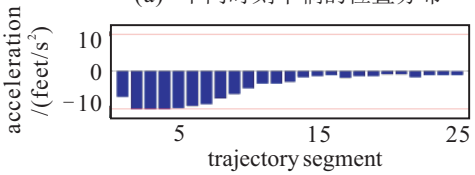
图 3 场景 1 中的轨迹规划结果

图 3(a) 和图 4(a) 中, 自上而下分别表示在 $t = 0\,\text{s}$, $t = 0.6\,\text{s}$, $t = 1.2\,\text{s}$ 和 $t = 1.8\,\text{s}$ 时刻所有车辆的位置分布, 曲线表示所生成的轨迹曲线, 圆圈表示轨迹起始点, 星花表示轨迹结束点, 深色矩形表示本车 EV, 白色矩形表示障碍物 α、β、γ. 图 3(b) 和图 4(b) 中给出了加速度在各轨迹片段的取值.

图 3(a) 中, 当前车道的前车 β 车速明显低于 EV, 右车道存在障碍物 α 阻挡 EV 进入, 而左车道存在换道空间. 因此, EV 的最优轨迹为换道至左车道 (即车道 C) 前进, 但需先减速以留出换道空间, 再加速进入目标车道 (见图 3(b)).



(a) 不同时刻车辆的位置分布



(b) 加速度在各轨迹片段的取值

图 4 场景 2 中的轨迹规划结果

图 4(a) 中, 如果周围障碍物按场景 2 运动 (除左车道的换道空间消失外, 其他条件不变), EV 的最优轨迹为留在当前车道 (即车道 B) 行驶, 但需减速以保持与前车的安全间距 (见图 4(b)).

3.2 粗糙评价分析

本节比较连接到相邻目标点的两条轨迹曲线的粗糙评价和精确评价. 在场景 1 中, 对目标点坐标分别为 (90,12) 和 (87,12) 的轨迹曲线 Γ_1 和 Γ_2, 各作 100 次粗糙评价实验和 1 次精确评价实验. 每次粗糙评价实验中, 记录求解 Π_c'' 的收敛过程, 并保存迭代次数 I_c 取不同值时得到的粗糙评价值. 100 次实验后, 统计不同 I_c 下得到的 100 个粗糙评价值的平均值 μ 和标准差 σ. 在精确评价实验中, 计算并记录得到的精确评价值. 图 5 用横轴表示粗糙求解的迭代次数, 纵轴表示评价值, 比较了轨迹曲线 Γ_1 和 Γ_2 的精确评价值和粗糙评价值. 其中, 星形线和三角线分别表示 Γ_1 和 Γ_2 的精确评价值, 点划线和圆圈线分别表示 Γ_1 和 Γ_2 的粗糙评价值, 上、下两块阴影区域分别表示 Γ_1 和 Γ_2 在不同 I_c 下 $(\mu - \sigma, \mu + \sigma)$ 对应的取值范围.

注意到, 在粗糙评价过程中, 当优化迭代至 $I_c = 15$ 时, 虽然所得评价值与精确评价值间存在偏差, 但仍然可以正确判断出 Γ_1 和 Γ_2 的优劣顺序. 用正态分布拟合 $I_c = 15$ 时 Γ_1 和 Γ_2 的粗糙评价值 X_1 和 X_2 的分布, 用 μ_1 和 σ_1 表示 X_1 的平均值和标准差, μ_2 和 σ_2

表示 X_2 的平均值和标准差, 则有 $X_1 \sim N(\mu_1, \sigma_1^2)$, $X_2 \sim N(\mu_2, \sigma_2^2)$, 可计算 Γ_2 优于 Γ_1 的概率为

$$P(X_2 < X_1) \approx 100\,\%. \tag{19}$$

由式(19)可知, 在概率意义上, OODE 算法能正确判断轨迹曲线 Γ_1 和 Γ_2 评价值的大小关系, 即能对 Γ_1 和 Γ_2 进行正确的择优和筛选. 因此, 引入粗糙评价后, OODE 算法虽然不能保证求得全局最优解, 但能保证所求解为概率意义上的全局满意解.

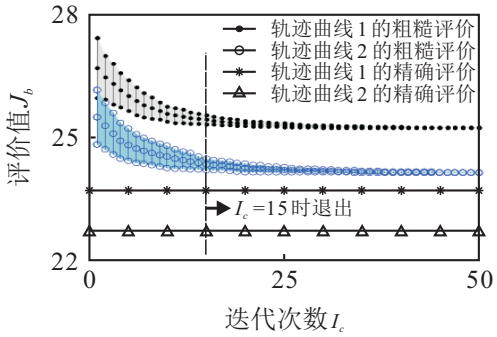


图 5　相邻目标点对应轨迹曲线的评价值比较

3.3　求解性能分析

本节对 OODE 算法、传统优化算法 TRA、DE 算法和传统轨迹规划方法 A* 的求解耗时和求解质量进行比较和分析. 与 OODE 算法不同的是, TRA 算法对每一条候选轨迹曲线都作精确评价, 再比较找出最优者. DE 算法并不预先生成候选轨迹曲线族, 而是将轨迹曲线参数向量 $\boldsymbol{b}$ 和加速度向量 $\boldsymbol{a}$ 放在同一优化问题 $\min\limits_{\boldsymbol{b},\boldsymbol{a}} J(\boldsymbol{b},\boldsymbol{a})$ (其中: $\boldsymbol{b} \in B^K, \boldsymbol{a} \in R^N$) 中直接求解. A* 算法在车辆的运动状态空间中考虑速度和加速度, 通过搜索找到最优轨迹. 上面 4 种算法完成场景 1 中轨迹规划的平均耗时 $\overline{T}$ 和平均最优轨迹评价值 $\overline{J(\boldsymbol{b}_{\rm opt}, \boldsymbol{a}_{\rm opt})}$ 如表 4 中所示.

表 4　平均耗时和平均最优轨迹评价值比较

	$\overline{T}$/ms	$\overline{J(\boldsymbol{b}_{\rm opt}, \boldsymbol{a}_{\rm opt})}$
OODE	95.7	20.20
TRA	496.8	20.20
DE	2 771.0	29.75
A*	756.2	20.55

OODE 算法与 TRA 算法的主要区别是前者引入了序优化思想, 通过粗糙评价轨迹曲线快速完成曲线的择优, 迅速缩小解空间的搜索范围, 而后者对全部曲线计算精确评价. 由表 4 可知, 与改进前相比, 平均求解耗时明显减少, 但平均最优轨迹评价值相同, 可认为序优化的引入大大提高了求解速度, 同时仍能保证求解质量.

DE 算法同时优化 $\boldsymbol{b}$ 和 $\boldsymbol{a}$, 虽然这样做是以求取全局最优解为目标, 但仿真结果表明 DE 算法的求解速度最慢, 且求解质量不如其余算法. 主要原因是, 一方面解空间的维度增加, 导致解的搜索范围明显增大, 迭代过程的收敛速度减慢; 另一方面, 求解器需要同时优化非线性目标函数和处理多个等式约束, 导致可行解空间的结构高度非凸, 使求解易陷入局部最优. 反观 OODE 算法虽然仅能保证求得概率意义上的满意解, 但其求解质量和效率反而更高.

A* 算法需要在高维度的车辆运动状态空间内不断评估新节点以探索车辆的运动域, 进而找到最优轨迹, 求解效率不高. 仿真结果表明, 虽然 A* 的求解质量接近 OODE 算法和 TRA 算法, 但求解速度更慢.

根据表 4, OODE 算法规划生成一条时间长度为 1.8 s 以上 (见图 3 和图 4) 的行驶轨迹的平均耗时为 95.7 ms, 其规划速度能满足实时应用的要求, 同时规划模型中障碍物的常速假设也是可行的.

表 5 给出了 OODE 算法完成一次轨迹曲线粗糙评价和精确评价的平均耗时 $\overline{T}_0$、轨迹划分片段数 N 和优化迭代次数 I.

表 5　轨迹曲线的粗糙评价与精确评价耗时比较

评价方式	平均耗时 $\overline{T}_0$/ms	轨迹片段数 N	迭代次数 I
粗糙评价	1.3	5	15
精确评价	83.0	25	100

对比粗糙和精确评价可知, 减小轨迹划分片段数 N 和降低优化迭代次数 I 时评价耗时明显减小. 由于粗糙评价的引入, 传统的 TRA 算法需要精确评价 G 次, 被改进为 OODE 算法只需要粗糙评价 G 次加精确评价 1 次. 用 T_c 和 T_a 分别表示进行 1 次粗糙评价和精确评价的耗时, 则改进前后的算法耗时比约为

$$\frac{T_{\rm TRA}}{T_{\rm OODE}} \approx \frac{G \cdot T_a}{G \cdot T_c + T_a}. \tag{20}$$

由式(20)可知, 场景越复杂(障碍物越多), 轨迹曲线采样精度越高(G 越大), 轨迹模型精度要求越高(N 越大), 引入粗糙评价对算法效率的改善越显著.

4　结　论

本文将智能汽车的驾驶决策和轨迹规划问题统一建模为 NLP 模型, 并基于序优化思想提出了一种新的混合智能优化算法 OODE. NLP 模型以最优化轨迹的综合性能(行驶效率、安全性、舒适性和经济性)为目标, 考虑车辆运动学约束, 规划生成最优的轨迹曲线和加速度序列. OODE 算法采用两层框架, 在外层和内层分别求解轨迹曲线和加速度序列, 通过粗糙评价轨迹曲线, 快速进行轨迹曲线的择优, 使得解空间的搜索范围迅速缩小, 保证了较高的求解效率. OODE 算法虽然不能保证求得全局最优解, 但能保证所求解为全局满意解. 仿真结果表明, 所提出的方法能够处理多障碍物的复杂交通场景, 求解速度能够满

足实时应用的需要. 此外, 本文中的规划方法是基于优化框架实现的, 因此具有一定的通用性, 不仅适用于智能汽车, 还能被扩展到机器人领域.

然而, 在已实现的OODE算法中, 粗糙评价的精度决定了最终解的精度. 研究粗糙评价与精确评价的联动关系, 轨迹求解质量还能提高, 这是未来进一步的研究内容.

参考文献(References)

[1] Coelingh E, Eidehall A, Bengtsson M. Collision warning with full auto brake and pedestrian detection-a practical example of automatic emergency braking[C]. The 13th Int IEEE Conf on Intelligent Transportation Systems(ITSC). Funchal: IEEE, 2010: 155-160.

[2] Zhenhai F A G, Liyong S B J. Optimal preview trajectory decision model of lane-keeping system with driver behavior simulation and artificial potential field[C]. 2009 IEEE Intelligent Vehicles Symposium. Xi'an: IEEE, 2009: 797-801.

[3] Kala R, Warwick K. Planning autonomous vehicles in the absence of speed lanes using lateral potentials[C]. 2012 IEEE Intelligent Vehicles Symposium. Alcala de Henares: IEEE, 2012: 597-602.

[4] Hilgert J, Hirsch K, Bertram T, et al. Emergency path planning for autonomous vehicles using elastic band theory[C]. Proc of 2003 IEEE/ASME Int Conf on Advanced Intelligent Mechatronics. Kobe: IEEE, 2003: 1390-1395.

[5] Crane Iii C D, Armstrong Ii D G, Touchton R, et al. Team CIMAR's NaviGator: An unmanned ground vehicle for the 2005 DARPA grand challenge[J]. J of Field Robotics, 2006, 23(8): 599-623.

[6] Urmson C, Anhalt J, Bagnell D, et al. Autonomous driving in urban environments: Boss and the urban challenge[J]. J of Field Robotics, 2008, 25(8): 425-466.

[7] Melchior N A, Simmons R. Particle RRT for path planning with uncertainty[C]. Proc of 2007 IEEE Int Conf on Robotics and Automation. Roma: IEEE, 2007: 1617-1624.

[8] McNaughton M, Urmson C, Dolan J M, et al. Motion planning for autonomous driving with a conformal spatiotemporal lattice[C]. 2011 IEEE Int Conf on Robotics and Automation(ICRA). Shanghai: IEEE, 2011: 4889-4895.

[9] Pivtoraiko M, Knepper R A, Kelly A. Differentially constrained mobile robot motion planning in state lattices[J]. J of Field Robotics, 2009, 26(3): 308-333.

[10] Papadimitriou I, Tomizuka M. Fast lane changing computations using polynomials[C]. Proc of the 2003 American Control Conf. Denver: IEEE, 2003: 48-53.

[11] Kanaris A, Kosmatopoulos E B, Loannou P A. Strategies and spacing requirements for lane changing and merging in automated highway systems[J]. IEEE Trans on Vehicular Technology, 2001, 50(6): 1568-1581.

[12] Jula H, Kosmatopoulos E B, Ioannou P A. Collision avoidance analysis for lane changing and merging[J]. IEEE Trans on Vehicular Technology, 2000, 49(6): 2295-2308.

[13] Kohler S, Schreiner B, Ronalter S, et al. Autonomous evasive maneuvers triggered by infrastructure-based detection of pedestrian intentions[C]. 2013 IEEE Intelligent Vehicles Symposium. Gold Coast: IEEE, 2013: 519-526.

[14] Madas D, Nosratinia M, Keshavarz M, et al. On path planning methods for automotive collision avoidance[C]. 2013 IEEE Intelligent Vehicles Symposium. Gold Coast: IEEE, 2013: 931-937.

[15] Montemerlo M, Becker J, Bhat S, et al. Junior: The stanford entry in the urban challenge[J]. J of Field Robotics, 2008, 25(9): 569-597.

[16] Ho Yu-Chi, Zhao Qian-Chuan, Jia Qing-Shan. Ordinal optimization: Soft optimization for hard problems[M]. New York: Springer, 2008: 7-8.

[17] Storn R, Price K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces[J]. J of global optimization, 1997, 11(4): 341-359.

[18] Miller I, Campbell M, Huttenlocher D, et al. Team Cornell's skynet: Robust perception and planning in an urban environment[J]. J of Field Robotics, 2008, 25(8): 493-527.

(责任编辑：齐 霁)