arXiv:2305.02621v2 [cs.RO] 9 Aug 2023

使用动态优化的城市环境实时空间轨迹规划

Jona Ruof、Max Bastian Mertens、Michael Buchholz 和 Klaus Dietmayer乌尔姆大学测量、控制与微技术研究所，德国{firstname}.{lastname}@uni-ulm.de

摘要——在城市环境中为自动驾驶车辆规划轨迹，需要具备高度通用性、较长规划时域和快速更新能力的方法。通过采用路径-速度分解，我们提出了一种新颖的规划框架，该框架能够生成具有前瞻性的轨迹，并有效处理各种状态约束和控制约束。与相关研究不同，所提出的最优控制问题是在空间域而非时间域内进行建模的。这种空间 formulation 将环境约束与优化变量解耦，从而能够应用简单而高效的打靶法。为此，我们在增广拉格朗日框架下，提出了一种基于 ILQR 的定制化求解策略，即使在初始解不可行的情况下，也能快速最小化轨迹目标代价。通过仿真评估以及在真实城市交通中使用一辆全尺寸自动驾驶车辆进行的测试，验证了所提出方法的实时运行能力和多功能性。

关键词—轨迹规划、自动驾驶、增广拉格朗日函数、迭代 LQR

I. 引言

轨迹规划是自动驾驶中众多应用的基础。近年来，相关研究日益聚焦于城市环境中的规划[1], [2], [3]。事实证明，这项任务颇具挑战，因为交通参与者之间的相互依赖以及由此产生的问题复杂性，往往使情形难以预测。为了在危急情况下仍能安全应对，必须高频计算轨迹更新。此外，城市交通情形的多样性带来了不同的空间、速度、加速度乃至时空约束，使得

需要具备高度通用性的规划方法。此外，所得轨迹的舒适性和安全性往往直接取决于所考虑的时域。因此，计算具有前瞻性的轨迹需要在 >10s 的时间区间内进行规划，或规划至车辆的感知范围（>100m）。通用性要求与长时域规划相结合，与快速反应时间相矛盾，因此构成了一个亟待解决的冲突。

针对上述两难问题，我们研究一种轨迹规划框架的构建方法，该框架旨在以高频率（例如 100Hz）运行，并且足够灵活，能够应对城市环境中的驾驶。与.....相比，

本研究的部分经费由德国联邦经济事务与气候行动部通过“复杂驾驶情境下的高度自动化与全自动驾驶”计划资助（LUKAS 项目，资助编号 19A20004F）。

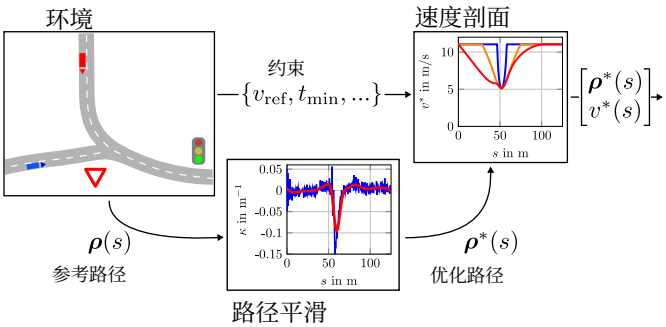

图 1. 给定一个示例城市交通场景时所提方法的概览。自车（蓝色）必须及时到达交通信号灯，同时遵守其他车辆（红色）的通行权。我们提出一种空间分解式路径—速度表述（第三节），并采用动态优化方法求解（第四节）。

与大多数相关研究一样，我们探索的是沿空间而非时间对轨迹进行优化。这一选择源于以下观察：在自动驾驶中，许多情境（例如限速、停车标志、交通信号灯等）施加的约束都是空间的函数 s。同样将轨迹定义为空间的函数后，约束值便与轨迹中由此产生的状态 $\boldsymbol{x}(s)$ 和控制变量 $\boldsymbol{u}(s)$ 脱钩。相比之下，在时间 formulation 中，例如，轨迹末端的 $s(t = 10\text{s})$ 取决于轨迹起始处加速度 $a(t = 0\text{ s})$ 的选择。对于局部固定的约束，空间 formulation 能够稳定优化，甚至可以使用高效的单次打靶方法（如 AL-ILQR）；这类方法通常被认为不太适用于较长的时间时域 [4]。

在结构上，我们提出采用路径-速度分解方法（图 1），本文重点关注纵向优化。我们的主要贡献如下：

- 一种动态优化建模方法，可在毫秒级时间内（<10 ms）求解，并生成覆盖 125m 时域的前瞻性轨迹。此外，该方法还考虑了静态和动态障碍物、速度、加速度、曲率限制以及时空约束。

- 一种基于增广拉格朗日框架中 ILQR 的定制化实时求解策略，具有更高的优化稳定性，并能隐式处理

© 2023 IEEE。允许个人使用本材料。所有其他用途均须获得电气电子工程师学会的许可，包括在任何当前或未来媒体中重印或再版本材料以用于广告或促销目的、创作新的合集作品、转售或重新分发至服务器或列表，或在其他作品中重复使用本作品中任何受版权保护的部分。数字对象标识符：10.1109/IV55152.2023.10186535

不可行初始化。

- 在仿真环境中以及真实城市交通中的全尺寸自动驾驶车辆上进行评估，展示了所提出方法的通用性和鲁棒性。
- 提供了以 C 和 Python 编写的开源实现以及仿真实验，相关内容可在 https://github.com/joruof/tpl 查看。

本文其余部分安排如下：在第二节讨论相关工作后，第三节介绍并阐述两个核心动态优化目标。随后，第四节概述求解策略，并在第五节通过仿真和真实世界实验评估所提出的方法。最后，第六节简要讨论当前的局限性和未来工作的方向。

II. 相关工作

与本文贡献相关的文献可分为三个主要类别，本节将对此进行讨论。

A. 城市轨迹规划框架

尽管已经开发出许多车辆轨迹规划策略，但大多数文献聚焦于特定场景，只有少数提出了适用于城市环境的通用方法。端到端学习方法（如 [5] ）展现出良好的结果，但通常考虑的时域较短，且存在泛化问题。近期， [1] 提出了一种基于 B´zier 采样的通用运动规划框架，并已证明其在真实世界场景中有效。Ziegler 等人 [6]提出了局部连续优化方法，在通用规划框架中生成轨迹，使车辆能够在城市公共道路上行驶。针对较长时域，Hubmann 等人 [7] 展示了将图搜索方法与连续优化相结合，进行纵向规划的方案。我们的前期工作 [2],与 [6], 类似，也依赖由驾驶员模型支持的连续优化，并且在城市公共道路上表现良好。另一项近期工作是 [3], ，它采用模型预测控制（MPC）的方式进行局部连续优化，并在仿真中展现了其在超车和切入场景中的良好效果。与我们的工作相比，上述所有文献要么以较低频率更新轨迹（ [7], [2] @ 10 Hz； [3], [1] @ 20 Hz； [6]在 0.5 s 后终止），要么仅考虑较短时域（ [3]中最长为 3 s），要么未处理显式时空约束。

B. 迭代线性二次调节器

迭代线性二次调节器（ILQR）算法 [8] 是一种单重射击、离散时间的局部连续轨迹优化方法。该方法已成功应用于各种机器人应用中的最优控制，但通常采用短时域，例如轨迹跟踪 [9], [10]。

已有进一步发展的 ILQR 变体，可用于处理例如多重射击目标函数 [4] 或控制约束 [11]。状态约束的处理方式包括内点变体 [12], 以及一种更简单的方法——约束迭代线性二次调节器（CILQR）；该方法由 [13] 提出，并在后续研究中得到进一步分析见 [14]。CILQR 使用障碍函数来施加状态约束，并已成功应用于自动驾驶：该方法最初在 [13], 中用于障碍物规避，之后扩展了随机预测和权重调节功能，以应对更多样的情形，例如车辆切入 [14]。此外， [15] 改善了 CILQR 中的数值问题，并引入了初始轨迹猜测；而 [16] 则利用 CILQR 进行规划，以降低加加速度，同时探索自动权重调节。

虽然 CILQR 中使用的障碍函数相对易于理解和实现，但在不可行初始化的情况下会出现数值问题（参见 [15]），并且必须手动调节，或采用自动权重调节方案进行调节（参见 [16], [14]）。最近，通过将 ILQR 纳入 ADMM（交替方向乘子法）方法 [17], 这些问题得到了处理。另一种在数值上更稳健但更简单的约束施加方法是增广拉格朗日（AL）方法，ALTRO 软件包 [18]将其实现为 AL-ILQR。尽管 AL-ILQR 在实现复杂度、运行时间复杂度和约束满足之间取得了平衡，但它尚未适用于自动驾驶中的长时域轨迹规划。

C. 空间轨迹规划

长时域空间轨迹规划的发展主要受到自动驾驶卡车运输领域研究的推动。早期研究（如 [19] ）已提出利用空间轨迹计算长时域轨迹，以降低燃油消耗，近期又在 [20] 中进行了扩展。与我们的工作不同，这些研究通过动态规划获取轨迹，且更新频率也低得多（例如 [20] 中为 200ms）。另一项近期研究 [21] 同样采用路径-速度分解，并在空间域中规划轨迹。与我们的在线方法不同，他们将部分规划问题离线求解，且不考虑时空约束。

III. 公式化

以下部分概述了在空间 中定义的一般路径目标和速度目标，并说明如何在目标函数中表示约束。

A. 路径目标

我们假设可以获取参考路径 $\boldsymbol{\rho}(s) = [\hat{x}(s),\ \hat{y}(s)]^T$，该路径是在二维笛卡尔坐标系中定义的空间向量值函数 s ，例如来自预先记录的 GPS地图或车道检测模块。由于路径曲率 $\kappa(s)$ 会直接限制最大速度，因此准确估计对于生成平滑的速度剖面至关重要。局部曲率计算会产生噪声结果 [22], ，因此我们采用 [22] 中的公式。由此，通过求解一个状态为 $\boldsymbol{x}_r = [x_r, y_r, \phi_r]^T$ 、控制量为 $u_r = \kappa$ 的最优控制问题，得到曲率经过平滑处理的路径。目标函数的动力学 $\partial \boldsymbol{x}_r / \partial s = \boldsymbol{f}_r(\boldsymbol{x}_r,\ u_r)$ 定义为

$$\frac{\partial x_r}{\partial s} = \cos(\phi_r)\,, \qquad \frac{\partial y_r}{\partial s} = \sin(\phi_r)\,, \qquad \frac{\partial \phi_r}{\partial s} = \kappa\,, \tag{1}$$

这些量由边界条件 $\boldsymbol{x}_r(0) = \boldsymbol{x}_{r,0}$ 沿空间 $s \in [0, S]$ 积分得到，其中 S 为路径时域。通过积分得到的轨迹采用二次函数进行评价（为便于阅读，省略 s ）

$$l_r(\boldsymbol{x}_r, u_r) = w_d \cdot ||(\hat{x} - x_r, \hat{y} - y_r)||_2^2 + w_\kappa \cdot ||\kappa||_2^2 , \quad (2)$$

其中， w_d， w_κ 分别是笛卡尔坐标误差和施加曲率的权重。加入盒约束 $\kappa \in [\kappa_{\min}, \kappa_{\max}]$ 以确保满足动力学可行性后，目标函数可表述为

$$\min_{\kappa(\cdot)} \int_0^S l_r(\boldsymbol{x}_r, u_r)\,\mathrm{d}s \quad \text{s.t.} \quad \frac{\partial \boldsymbol{x}_r}{\partial s_r} = \boldsymbol{f}_r(\boldsymbol{x}_r, u_r)\,, \quad (3a)$$

$$\boldsymbol{x}_r(0) = \boldsymbol{x}_{r,0}\,, \qquad \kappa_{\min} \le \kappa \le \kappa_{\max}\,. \quad (3b)$$

对于目标函数的极小化解 $\kappa^*(s)$，通过分别对 x_r、 y_r 关于 s 积分，即可得到曲率经过平滑处理的可行驶参考路径 $\boldsymbol{\rho}^*(s)$。

B. 速度目标函数

从 $a(t) = \partial v/\partial t$ 出发，应用链式法则即可定义空间域速度动力学

$$a(t) = \frac{\partial v}{\partial t} = \frac{\partial v}{\partial s}\frac{\partial s}{\partial t} = \frac{\partial v}{\partial s} v(t) \quad \Leftrightarrow \quad \frac{\partial v}{\partial s} = \frac{a(t)}{v(t)}\,. \quad (4)$$

类似地，加加速度 j 和时间 t 的空间域动力学为 $\partial j/\partial s = j(t)/v(t)$、 $\partial t/\partial s = 1/v(t)$。这一动力学函数的一个明显问题是在 $v = 0$ 处存在奇点。通过限制 $v \ge v_{\min}$ 可避免数值问题，其中 $v_{\min}$ 是一个很小的正值。如果目标速度 $v_{\lim} = 0$，则在后处理步骤中将 $v_{\min}$ 处的速度设为 0。一般而言，我们假设目标/最大速度在空间上的定义为 $v_{\lim}(s)$。下一小节将详细介绍如何根据具体交通状况生成 $v_{\lim}(s)$。根据 $v_{\lim}$ ，我们生成参考轨迹 v_{ref} 和速度剖面 v^*，如图 2 所示。下面将说明这一轨迹生成过程。

由于 $v_{\lim}(s)$ 可能不满足加加速度或加速度约束$[j_{\min}, j_{\max}]$, $[a_{\min}, a_{\max}]$，因此首先通过积分对约束曲线进行塑形

$$\left(\frac{\partial v}{\partial s}, \frac{\partial a}{\partial s}\right)^T = \begin{cases} (a/v,\, j/v)^T & : v \le v_{\lim} \\ (0, 0)^T & : \text{else} \end{cases} \quad (5)$$

分别从时域 S 反向积分至 0，此时 $j(s) = j_{\min}$，边界条件为 $(v(S)\ a(S))^T = (v_{\lim}(S), 0)$；以及正向积分，此时 $j(s) = j_{\max}$，边界条件为 $(v(0)\ a(0))^T = (v(0), 0)$。此外，在对生成的曲线 v_{bwd} 和 v_{fwd}进行积分时，每个积分步之后，状态 v, a 均限制为 $v = \min(v, v_{\lim})$ 和 $a = \text{clip}(a_{\min}, a_{\max}, a)$。应用最小值运算符后，得到一条曲线

$$v_{\text{ref}}(s) = \min(v_{\text{bwd}}(s), v_{\text{fwd}}(s))\,, \quad (6)$$

它遵守加速度和加加速度限制，但既缺乏前瞻性，也不够舒适。因此， v_{ref} 随后被纳入动态优化目标函数，作为速度目标以及可行速度曲线的上限函数。

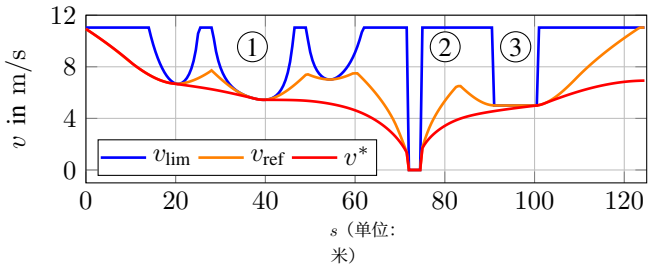

图 2. 速度剖面生成的中间步骤：基于 $v_{\lim}$，生成一个受加速度和加加速度限制的速度剖面 v_{ref} ，随后对其进行优化，以获得具有前瞻性的 v^* 速度剖面。标记 1 展示了一个由道路曲率引起的速度约束示例。标记 2 和 3 展示了交互约束。

为此，我们定义 $\boldsymbol{x}_v = [v, t]^T$、 $u_v = a$ 和$f_v(\boldsymbol{x}_v, u_v) = [\partial v/\partial s, \partial t/\partial s]^T$ ，其边界条件为$v(0) = v_{\text{veh}}$。通过对 $f_v(\boldsymbol{x}_v, u_v)$ 进行正向积分生成的轮廓 $v(s)$，由二次函数进行评估。

$$l_v(\boldsymbol{x}_v, u_v) = w_v||v(s) - v_{\text{ref}}(s)||_2^2 + w_a||a(s)||_2^2 \quad (7)$$

权重为 w_v、 w_a。结合动力学 f_v 和式 (7)，可得到关于速度剖面的动态最小化目标函数

$$\min_{a(\cdot)} \int_0^S l_v(\boldsymbol{x}_v, u_v)\,\mathrm{d}s \quad \text{s.t.} \quad \frac{\partial \boldsymbol{x}_v}{\partial s} = f_v(\boldsymbol{x}_v, u_v)\,, \quad (8a)$$

$$v_{\min} \le v \le v_{\text{ref}}\,, \qquad a_{\min} \le a \le a_{\max}\,, \quad (8b)$$

其中，最优速度剖面 $v^*(s)$ 是通过对该目标函数的最小化解 a^* 进行积分，并利用动力学方程 (4) 得到的。

尽管代价函数 l_v 采用了简单的二次形式，但由于非线性动力学以及控制约束和状态约束 (8b) 的存在，目标函数变得复杂。因此，需要一种专用算法来高效求解该目标函数，我们将在第四节中提出该算法。

C. 限速

在城市自动驾驶场景中，速度限制函数 $v_{\lim}(s)$（图 2）由法定限速 v_{lg} 以及道路曲率 $\kappa^*(s)$ 所引起的隐式速度限制共同决定。给定允许的最大横向加速度 $\hat{a}_{\text{lat}}$， $v_{\lim}$ 初始定义为

$$v_{\lim}(s) = \min\left(v_{\text{lg}}(s), \sqrt{|\hat{a}_{\text{lat}}|/|\kappa^*(s)|}\right)\,, \quad (9)$$

其中， $\kappa^*(s)$ 是通过求解目标函数 (3) 得到的。

D. 交互约束

静态交互约束固定在某个位置 s_o，只需使用最小值运算符更新 $v_{\lim}$ 即可表示。

$$v_{\lim}(s_o) \leftarrow \min(v_{\lim}(s_o), v_o)\,, \quad (10)$$

其中， $v_o = 0$ 适用于需要停车的交互（例如交通信号灯）， $0 \le v_o \le v_{\lim}$ 适用于速度降低的交互。请注意，通过公式 (5) 中介绍的反向-正向积分，即使 $v_{\lim}$ 发生显著变化（例如经过障碍物），最终仍会得到平滑的参考轨迹 v_{ref} （图 2）。

与静态交互类似，我们将位置为 s_o 、速度为 v_o 的通用动态交互（例如汽车、自行车等）视为速度剖面中的约束。障碍物通过反应式速度约束 formulation 表示，其依据是恒速预测。给定安全距离 d_{safe} （该距离可能取决于 v），约束会相应地更新 v_{lim} 】【。

$$v_{\text{lim}}(s) \leftarrow \min(v_{\text{lim}}(s), v_o \cdot \min(1, s_o/d_{\text{safe}}))\,, \qquad (11)$$

在所有 $s \in [s_o - d_{\text{safe}}, s_o]$ 中。由此产生的速度约束确保 $v = v_o$ 位于安全距离的起点处，然后使 v 按照安全距离违规量成比例地降低，直至在 $s = s_o$处进入静止状态。这种启发式方法确保，在存在减速障碍物的情况下，只要障碍物的减速度不超过 a_{min}，距离就会增加并趋近于安全距离。即使前方存在缓慢障碍物，只要在相同动力学限制下存在可行解，也能够避免碰撞。尽管这种处理障碍物的方式看似简单，但在采用快速重规划周期的前提下，它能够产生安全且平滑的行为。

E. 时空约束

时空约束（如图 1 所示）将自车限制在某一位置的特定时间区间内。最大时空约束要求车辆在时间 $t(s) \leq t_{\max}(s)$ 到达点 s，这可能会增加加速度和速度。只要速度保持低于 v_{ref} 且加速度处于 $[a_{\text{min}}, a_{\text{max}}]$以内，该约束始终可行。因此，可以将 $t_{\max}(s)$ 约束简单地作为目标函数（8）中的附加状态约束。相比之下，最小时空约束要求 $t(s) \geq t_{\text{min}}(s)$，这可能会使速度降低至静止状态。直接应用 t_{min} 约束而不进行额外正则化，通常会导致车辆在远离约束位置的地方停车。虽然这种解是有效的，但其他交通参与者通常会认为它不直观或具有阻碍性。因此，我们提出通过以下不等式，在 s_c 处表示一个 t_{min} 约束

$$(t_{\text{min}}(s_c) - t(s_c)) \cdot (v(s_c) - v_{\text{min}}) \leq 0\,. \qquad (12)$$

左侧在第一个因子中原样包含约束，而在第二个因子中通过将速度降至 v_{min} 也趋近于零。因此，不等式（12）以连续的方式表达了：缩短在 s_c 至 t_{min} 的时间以及在 s_c 处停车，均可实现约束满足。此外，我们将式（7）中参考速度偏差 w_v的权重替换为由 s塑形的 w_v,

$$w_v(s) = \min(1, ((s - s_c - \alpha) \cdot \beta)^2)\,. \qquad (13)$$

对于给定的偏移量 $\alpha \geq 0$ 和缩放系数 $0 < \beta < 1$，该加权函数缓解了以 v_{ref} 行驶与降低 v之间的冲突，从而满足 $t_{\text{min}}(s_c)$。由于该函数在 s_c处及以上取最小值，因此会促使车辆恰好在 s_c处停车。在我们的实验中，将 (12)和 (13)同时纳入目标函数 (8)，会产生明确的约束处理行为：要么在约束前减速，要么恰好在 s_c处停车。

算法 1求解策略

将 $n \leftarrow 1$ 和 μ_i 初始化为固定的大值
初始化 $\boldsymbol{X}$，$\boldsymbol{U}$，$\boldsymbol{\lambda}$，并从上一步平移而来
当 $n < N$ **且**代价变化 $>$ tol **时执行**
通过 ILQR 迭代最小化目标函数（14）
结束循环
$\lambda_{k,\,i} \leftarrow \lambda_{k,\,i} + \mu_i h_i\,(\boldsymbol{x}_k, \boldsymbol{u}_k)$
$\lambda_{k,\,i} \leftarrow \max(0,\ \min(\lambda_{k,\,i},\ \lambda_{\max,\,i}))$

IV. 求解策略

该求解策略基于 ILQR。ILQR 是一种单次打靶、离散时间、局部连续的轨迹优化方法，用于具有一般非线性动力学和代价函数的无约束动态优化目标（参见 [8]）。我们认为，ILQR 适用于汽车应用。首先，对于较小的控制空间，它可以用纯 C 代码高效实现（例如在嵌入式设备上），甚至无需专门的数值计算例程。其次，ILQR 的运行时间复杂度仅随优化时域线性增长，因此适用于长时域规划。第三，作为一种打靶法，ILQR 生成的轨迹在设计上始终具有动力学一致性，即使提前终止优化也不例外。这使得我们可以用反应时间换取暂时的次优解，并在后续多次迭代中逐步改进该解（收敛结果参见 [23]）。

A. 增广拉格朗日迭代线性二次调节

如 [18], 所示，ILQR 可以嵌入增广拉格朗日法中，以求解离散化的受约束动态优化目标函数。

$$\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_K} \mathcal{L}(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \quad \text{s.t} \quad \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \qquad (14)$$
$$h_i(\boldsymbol{x}_k, \boldsymbol{u}_k) \leq 0\,, i \in [1, ..., I],\, \forall k \in [1, ..., K]\,,$$

其中，$\boldsymbol{X} = (\boldsymbol{x}_1,\ ...,\ \boldsymbol{x}_K)$、$\boldsymbol{U} = (\boldsymbol{u}_1,\ ...,\ \boldsymbol{u}_K)$分别是状态矩阵和控制量矩阵，$\boldsymbol{\lambda} \in \mathbb{R}^{K \times I}$ 是拉格朗日乘子估计值矩阵，$\boldsymbol{\mu} \in \mathbb{R}^I$ 是障碍权重向量，f是动力学函数，h_i 是 I 个约束函数，k是求和指标，其上限为时域 K。此外，$\mathcal{L}$ 表示拉格朗日函数

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \qquad (15)$$
$$\sum_{k=1}^{K} \left[l(\boldsymbol{x}_k, \boldsymbol{u}_k) + \sum_{i=1}^{I} \left[\lambda_{k,i} h_i(\boldsymbol{x}_k, \boldsymbol{u}_k) + \mu_i h_i(\boldsymbol{x}_k, \boldsymbol{u}_k)^2 \right] \right],$$

其中 l 是积分代价。随后，增广拉格朗日迭代线性二次调节（AL-ILQR）交替执行以下步骤：使用标准 ILQR 求解目标函数 (14)，直至达到给定容差；按加权约束违反量增加拉格朗日乘子；并单调增加二次障碍权重，直至约束满足容差要求（参见 [18]）。

表I 评估中使用的目标参数

参数	取值
$v_{\min}$	1.0 米/秒
$a_{\min}, a_{\max}, \hat{a}_{\text{lat}}$	±2.5米/秒^2, 2.5 米/秒^2
$j_{\min}$, $j_{\max}$	±1.5 米/秒^3
$\kappa_{\min}$, $\kappa_{\max}$	±3.0 米$^{-1}$
w_d, w_κ, w_v, w_a	1.0, 20.0, 0.1, 1.0
α, β	10.0, $5 \cdot 10^{-3}$

B. 用于汽车轨迹规划的 *AL-ILQR*

[18] 中的 AL-ILQR 方法主要面向机器人应用，在这些应用中，将约束违反控制在较小范围内至关重要。将 AL-ILQR 用于汽车轨迹规划时，我们认为反应时间更为关键，而短暂的微小数值约束违反是可以容忍的。算法 1 列出了调整后的求解策略。由于算法 1 以较高频率运行，我们发现，只需进行一次 $\boldsymbol{\lambda}$ 更新，并采用较大的初始 $\boldsymbol{\mu}$，就能充分提高约束满足程度，从而避免多次增广拉格朗日迭代。总之，我们提出以下修改：

（i）障碍权重 μ_i 以较大值初始化并保持不变。这会降低最终的约束满足程度，但可避免障碍权重过高时出现数值病态。此外，这样可以使用标准的黎卡提反向传递，并避免平方根 LQR 反向传递带来的更高实现复杂度和运行时间（参见 [18]）。

（ii）拉格朗日乘子 $\lambda_{k,\,i}$ 被限制在 $\lambda_{k,\,i} \leq \lambda_{\max,\,i}$范围内，因此同样可以避免数值问题。在许多情况下，这甚至允许稳健地处理不可行的初始化。对于我们的应用而言，由受限 $\boldsymbol{\lambda}$ 导致的约束违反，优于完全无法获得解。

（iii）我们不像 [18], 中那样执行解精修，因为这会以增加运行时间为代价来降低约束违反程度。不过，在我们的情况下，更快的更新更为重要。

（iv）除第一个优化步骤外，我们采用 MPC 方式重新初始化 AL-ILQR，并将其上一步的结果沿车辆已行驶的距离进行平移。重要的是，这也包括 $\boldsymbol{\lambda}$，从而可以在规划步骤之间保留拉格朗日乘子。这样可以避免在每个规划步骤中完全重新计算 $\boldsymbol{\lambda}$，进而减轻对多次增广拉格朗日迭代的需求。

V. 结果

为了研究所提出方法的有效性，我们在模拟场景和真实世界场景中对其进行测试，这些场景展现了城市环境中的典型情况。

A. 实现

所提出的目标函数和求解策略以独立的 C 代码实现，并作为 Python 扩展。导数通过 SymPy 软件包进行符号微分获得。后续处理步骤则用 Python 实现。我们假设使用感知系统栈，为轨迹规划算法生成所需信息。在两种场景中，参考线 $\boldsymbol{\rho}(s)$

算法 2实时空间轨迹规划

1：给定自车状态、参考线 $\boldsymbol{\rho}$、环境状态
2：$\boldsymbol{\rho}^* \leftarrow$ 使用算法 1 最小化目标函数 (3)
3：提取与 $\boldsymbol{\rho}^*$相关的障碍物
4：将 $\boldsymbol{\rho}^*$, κ^*和障碍物转换为约束 $v_{\lim}$、$t_{\min}$、$t_{\max}$
5：生成 v_{ref} （第二节 III-B）
6：使用算法 1 最小化目标函数 (8) $v^* \leftarrow$
7：通过低级控制器执行（$\boldsymbol{\rho}^*$, v^*）

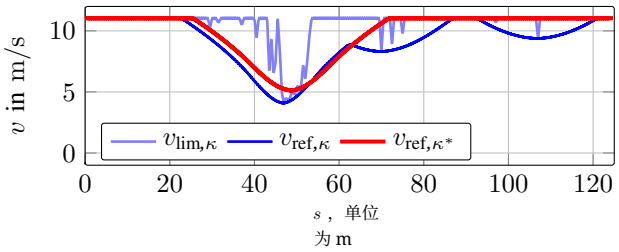

图3. 路径目标 (3) 的效果：使用局部曲率估计 κ会导致噪声较大的 $v_{\lim}$ 和不必要的剧烈加加速度 v_{ref}，而 κ^* 则能生成平滑的参考轨迹。

由在真实道路上采集的噪声GPS数据提供，因此需要进行第III-A节所述的路径优化。实现的所有步骤总结于算法2中，其中低级控制器的周期时间为 10 ms。如果规划步骤的运行时间超过 10 ms，控制器将执行上一次的解。目标函数中的积分必须进行离散化。对于所有场景，我们采用 $\Delta s = 0.5$m的离散步长，并选择 $S = 125$m，从而使每个目标函数包含 250 个空间步长。如果相对代价变化为 $< 10^{-6}$，或达到 $N = 5$ 次ILQR迭代，则终止两种目标函数最小化，以确保实时可行性。对于 $t_{\max}$ 约束，采用障碍权重和乘子限值 $\mu = 10^3$, $\lambda_{\max} = 10^3$；对于其他所有约束，则采用 $\mu = 10^2$, $\lambda_{\max} = 10^2$。其他参数通过实验确定，以实现舒适的行为（表I）。

B. 仿真场景

仿真实验在 AMD 1900X CPU @ 3.8GHz 基准时钟下以单线程执行。假设能够完美获知所有车辆的状态。车辆运动采用运动学自行车模型进行仿真。本文研究图1所示的复杂场景。其他场景可参见项目网站 https://github.com/joruof/tpl。

图1展示了 $t = 0$ 处的场景设置。自车必须在不停车的情况下平稳并入主干道，同时让行给另一辆以限速 $v_{\text{lg}} = 40$km/h 行驶的交通参与者。另一个复杂因素是，车辆通过交叉路口后还必须通过一盏交通信号灯，该信号灯计划在 $t = 14$s 时变红。图3显示，第二节中的路径优化可实现精确的曲率估计，并生成平滑的v_{ref} 曲线。基于优化路径，可生成两个时空

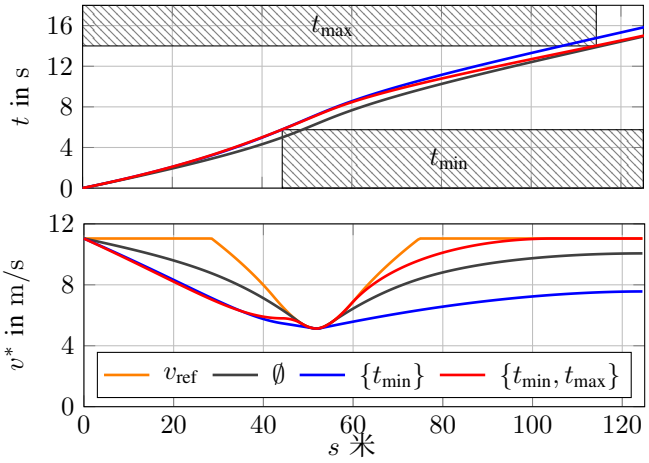

图4. 图1所示模拟场景中，在不同的时空约束（无约束、仅 $t_{\min}$、 $t_{\min}$ 和 $t_{\max}$）下， $t = 0$ 处的速度剖面以及行驶的 s-t 轨迹。时空约束以阴影矩形表示。

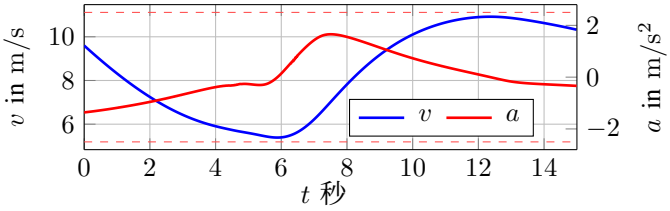

图5. 图1所示模拟场景中自车的速度和加速度（其中 $a_{\min}, a_{\max}$ 以虚线表示）。

表II 图1所示模拟场景中我们方法的运行时间。

组成部分	Mean	标准差	Max.
横向（公式（3））	0.354 毫秒	0.116 毫秒	1.50 毫秒
纵向（公式（8））	0.214 毫秒	0.101 毫秒	1.380 毫秒
总计（纬度 + 经度 + 预处理）	1。4 毫秒	0.290 毫秒	9.610 毫秒

约束 $t_{\min}(44.5\text{m}) = 5.75\text{s}$、 $t_{\max}(114.5\text{m}) = 14.0\,\text{s}$。图4比较了不同的时空约束集所产生的速度曲线。若没有时空约束（$\emptyset$），车辆会过早到达交叉路口，从而导致碰撞。仅应用 $\{t_{\min}\}$ 时，车辆可安全驶入交叉路口，但会错过绿灯。应用$\{t_{\min}, t_{\max}\}$ 后，系统找到一个解：首先在 44.5m 处降低速度以满足 $t_{\min}$ 约束，然后紧跟参考速度 v_{ref}，及时到达交通信号灯处。如图4和图5所示，该解满足所有加速度、速度和时空约束。表II给出了我们的方法在仿真场景中的运行时间统计。所提方法的总运行时间（包括预处理）最长不超过 10 ms。结果表明，采用所提方法能够稳健且高效地处理多个相互冲突的约束。

C. 真实世界场景

真实世界场景使用本研究所的一辆自动化测试车辆进行，该车辆为一辆配备了

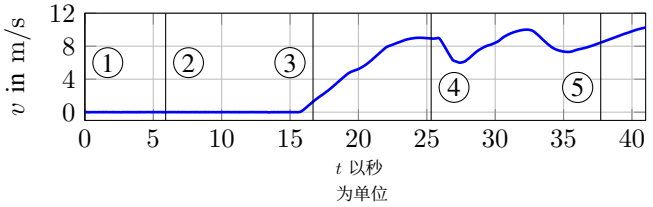

图 6. 真实世界场景中的车辆速度。事件 1—5 已用数字标记，并在图 7 中进行了详细说明。

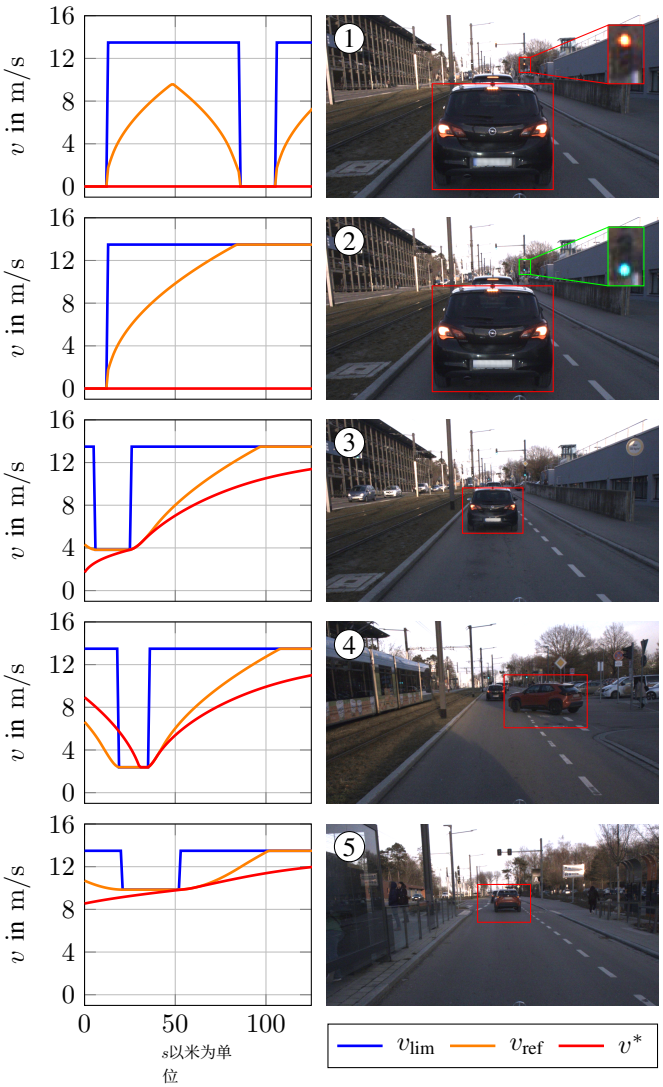

图 7. 左列展示了真实世界场景中事件 1–5（图 6）的速度剖面优化结果，其中 $s = 0$ 是自车各自的当前位置。右列展示了自车前置摄像头拍摄的图像供参考，相关物体均用方框标出。

配备了额外的传感器（例如摄像头、激光雷达）。转向和制动执行器以及传动系统均由我们的规划方法控制。车辆状态由厘米级精度的差分全球定位系统（DGPS）单元获取。所有计算均由一台搭载 AMD 3990X CPU（@ 2.9GHz 基准时钟）的车载 PC 执行，我们的规划器使用单线程运行。

表 III 规划器在真实世界场景中的运行时间。

组件	Mean	标准差。	Max.
横向（式（3））	0.529 毫秒	0.195 毫秒	1.371 毫秒
纵向（式（8））	0.924 毫秒	0.335 毫秒	2.371 毫秒
总计（横向 + 纵向 + 预处理）	6.796 毫秒	1.627 毫秒	14.15 毫秒

作为真实世界场景，展示了在密集城市环境中的规划。自车必须沿着参考线行驶，同时对环境和其他交通参与者作出反应。图 6 显示了记录的车辆速度，其中标记了 5 个事件。车辆在这些事件中的相应速度剖面与前视摄像头图像一同显示于图 7 中。

事件 *1*：起初，自车停在另一辆车后方，在红灯前等待。生成的速度剖面显示了两段速度为 0 的区间，分别对应前车和交通信号灯。

事件 *2*：在 $t = 5.9$ 秒时，交通信号灯变为绿灯，第二段 $v = 0$ 随之消失。由于前车尚未开始移动，速度剖面起始时的速度仍为 0。

事件 *3*：随着前车开始移动，$v \leq v_{\text{ref}}$ 约束逐渐解除，使优化能够提高速度，从而实现平滑跟驰行为。

事件 *4*：一辆汽车意外地切入自车前方，导致解算结果低于安全距离，从而违反速度约束。在这种情况下，障碍函数约束 [13]会迅速使代价升至极大值（或未定义），导致优化在数值上不稳定 [14]。由于 $\boldsymbol{\lambda}$，$\boldsymbol{\mu}$ 所提出的限制，我们调整后的求解策略（参见第四节）保持了数值稳定，并能够做出恰当反应。解算结果 v^* 会施加最小加速度，直至满足约束，同时保持轨迹其余部分的一致性。

事件 *5*：切入后，自车继续正常跟随前车，直至场景结束。

表 III 列出了我们的方法在真实世界场景中记录的运行时间。由于需要预处理的目标数量较多，总计算时间高于仿真场景，最高超过 10 毫秒。不过，97.4% 的规划步骤耗时不到 10 毫秒，这表明即使在复杂的真实世界场景中，我们的方法也能快速提供轨迹更新。

VI. 结论

我们提出了一种用于前瞻性轨迹规划的动态优化方法，能够处理一般速度以及时空约束。通过采用空间 formulation，可以应用快速且针对性设计的 AL-ILQR 方法，从而缩短循环时间，并显著降低复杂城市环境中的反应时间。由于所提出的 formulation 目前仅沿给定的参考线进行纵向规划，未来工作将着重于扩展方法，使其能够处理横向偏差（例如超车），并进一步降低最大计算时间。

参考文献

[1] A. Artunedo、J. Villagra 和 J. Godoy，《城市环境下自动驾驶的实时运动规划方法》，IEEE Access，第 7 卷，第 180039–180053 页，2019 年。[2] M. J. Graf、O. M. Speidel、J. O. Ruof 和 K. Dietmayer，《乌尔姆大学自动驾驶车辆的道路运动规划》，IEEE 智能交通系统杂志，第 14 卷，第 121–131 页，2022 年。[3] F. Micheli、M. Bersani、S. Arrigoni、F. Braghin 和 F. Cheli，《面向城市自动驾驶的 NMPC 轨迹规划器》，车辆系统动力学，第 1–23 页，2022 年。[4] M. Giftthaler、M. Neunert、M. Stauble、J. Buchli 和 M. Diehl，《用于非线性最优控制的一类迭代高斯-牛顿射击方法》，电气电子工程师学会智能机器人与系统国际会议，第 6903–6910 页，2018 年。[5] S. Casas、A. Sadat 和 R. Urtasun，《MP3：一种用于地图构建、感知、预测和规划的统一模型》，载于电气电子工程师学会/计算机视觉基金会计算机视觉与模式识别会议论文集，第 14403–14412 页，2021 年。[6] J. Ziegler、P. Bender、T. Dang 和 C. Stiller，《伯莎的轨迹规划：一种局部、连续的方法》，电气电子工程师学会智能车辆研讨会论文集，第 450–457 页，2014 年。[7] C. Hubmann、M. Aeberhard 和 C. Stiller，《适用于城市环境的通用驾驶策略》，电气电子工程师学会智能交通系统会议论文集，第 1010–1016 页，2016 年。[8] Y. Tassa、T. Erez 和 E. Todorov，《通过在线轨迹优化综合并稳定复杂行为》，电气电子工程师学会智能机器人与系统国际会议，第 4906–4913 页，2012 年。[9] A. Nagariya 和 S. Saripalli，《利用神经网络动力学模型为越野和公路车辆设计的迭代 LQR 控制器》，电气电子工程师学会智能车辆研讨会论文集，第 1740–1745 页，2020 年。[10] J. Chen、C. Liu 和 M. Tomizuka，《Foad：基于快速优化的自动驾驶运动规划器》，美国控制会议论文集，第 6 卷，第 4725–4732 页，2018 年。[11] Y. Tassa、N. Mansard 和 E. Todorov，《控制受限微分动态规划》，电气电子工程师学会机器人与自动化国际会议论文集，第 1168–1175 页，2014 年。[12] A. Pavlov、I. Shames 和 C. Manzie，《内点微分动态规划》，IEEE 控制系统技术汇刊，第 29 卷，第 2720–2727 页，2021 年。[13] J. Chen、W. Zhan 和 M. Tomizuka，《面向道路自动驾驶运动规划的约束迭代 LQR》，电气电子工程师学会智能交通系统会议论文集，第 3 卷，第 1–7 页，2018 年。[14] J. Chen、W. Zhan 和 M. Tomizuka，《采用约束迭代 LQR 的自动驾驶运动规划》，IEEE 智能车辆汇刊，第 4 卷，第 244–254 页，2019 年。[15] Y. Shimizu、S. Kato、W. Zhan、L. Sun、J. Chen 和 M. Tomizuka，《采用扩展约束迭代 LQR 的自动驾驶运动规划》，动态系统与控制会议，第 1 卷，2021 年。[16] O. Jahanmahin、Q. Lin、Y. Pan 和 J. M. Dolan，《用于双车道道路类人驾驶的最小加加速度 CILQR》，电气电子工程师学会智能车辆研讨会论文集，第 7 卷，第 1282–1289 页，2021 年。[17] J. Ma 、 Z. Cheng 、 X. Zhang 、 M. Tomizuka 和 T. H. Lee，《用于自动驾驶约束迭代 LQR 的交替方向乘子法》，IEEE 智能交通系统汇刊，第 1–12 页，2022 年。[18] T. A. Howell、B. E. Jackson 和 Z. Manchester，《ALTRO：一种用于约束轨迹优化的快速求解器》，电气电子工程师学会智能机器人与系统国际会议，第 7674–7679 页，2019 年。[19] E. Hellstr¨om、M. Ivarsson、J. ˚Aslund 和 L. Nielsen，《重型卡车前瞻控制：缩短行程时间并降低燃油消耗》，控制工程实践，第 17 卷，第 245–254 页，2009 年。[20] S. Bae、Y. Kim、J. Guanetti、F. Borrelli 和 S. Moura，《面向自动驾驶并与交通信号灯通信的生态自适应巡航控制设计与实现》，美国控制会议论文集，第 4628–4634 页，2019 年 7 月。[21] R. Voswinkel、I. Mutlu、K. Alaa 和 F. Schrodel，《一种用于自动驾驶的模块化无模型轨迹规划策略》，欧洲控制会议，第 1186–1191 页，2020 年。[22] F. Gritschneder、K. Graichen 和 K. Dietmayer，《利用基于梯度的非线性模型预测控制进行自动驾驶车辆快速轨迹规划》，电气电子工程师学会智能机器人与系统国际会议，第 7369–7374 页，2018 年。[23] K. Graichen 和 A. Kugi，《无终端约束的次优 MPC 的稳定性与渐进改进》，IEEE 自动控制汇刊，第 55 卷，第 2576–2580 页，2010 年。