arXiv:2303.00981v1 [cs.RO] 2 Mar 2023

利用插值径向基函数网络为类汽车机器人生成可微轨迹

Hongrui Zheng、Rahul Mangharam

摘要——自动驾驶车辆软件的设计大多沿用了感知-规划-行动 模型。传统的模块化自动驾驶车辆软件栈会分别开发感知、规划和控制软件，在针对不同目标进行优化时几乎没有集成。另一方面，端到端方法通常缺乏基于模型的白盒规划和控制策略所提供的原理。我们提出一种计算效率高的方法，利用插值径向基函数网络近似闭式轨迹生成，从而在这两种方法之间开辟一条折中路线。该方法能够平滑地近似从参数化优化问题到可行解的局部利普希茨连续映射。我们证明，这种可微近似的计算效率很高，并且在用作规划策略时，能够与感知和控制算法实现更紧密的集成。

一、引言

传统上，类汽车机器人的运动规划任务需要在线合成轨迹。局部规划任务最终需要搜索一组可行控制输入，以实现给定的局部目标。这一任务自然可以表示为一个参数化优化问题。车辆动力学约束或运行范围限制都很容易纳入其中。然而，即使使用高效的优化求解器，每秒最多三十次在线求解数百个具有不同规格的优化问题，仍然颇具挑战。人们曾使用大型查找表来存储通过离线运行优化得到的离散化解，以加快在线处理速度。然而，查找表只能提供实际最优控制的离散近似。此外，若没有专门的软件实现，轨迹生成仍然是单线程的，且内存占用很高。

近年来，不同的研究工作尝试通过创建端到端规划器来绕过在线求解最优控制，这些规划器旨在利用传感器信息 [1]–[3]直接生成控制输入。端到端方法的优势在于，可为上游和下游处理流程（例如，基于神经网络的感知流程）提供梯度信息。尽管这些方法展现出

尽管在高效生成局部运动规划方面具有潜力，但如果没有外部辅助，它们无法施加动力学约束，也无法保证解的有效性。我们提出插值径向基函数网络 （IRBFN），这是一种可微轨迹生成方法，能够生成满足动力学约束的轨迹，并可高效地进行并行化。不同于现有的可微规划器，从局部目标到轨迹终止状态的整个过程中都能获得梯度信息，

所有作者均来自美国宾夕法尼亚州费城 19104，宾夕法尼亚大学电气与系统工程系。电子邮箱：{hongruiz, rahulm}@seas.upenn.edu

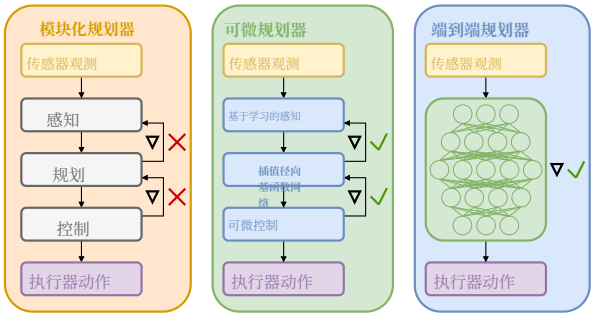

图 1. 标准模块化规划器栈、可微规划器栈与端到端规划器栈的比较。传统模块化规划器的各模块之间无法获取梯度信息，而可微规划器和端到端规划器则可以。

梯度信息。我们的方法还利用图形处理器执行高度可并行化的计算，从而提高效率。我们工作的一个关键贡献是：只要提供足够的训练样本，该方法就能保留动力学约束，并以任意高的精度逼近最优控制问题的解。与现有工作相比，另一个重要贡献是，该规划器相对于所有可能的参数都是可微的，例如，梯度可以从控制或规划损失流向局部目标选择。本文仅包含一项理论贡献，旨在支持可微规划器的使用，并提供一个重要的误差界。如图 1 所示，可微规划器融合了传统模块化规划器的可解释性与端到端规划器的可扩展性。

在以下各节中，我们将在第 III-A 节介绍有关轨迹生成和径向基函数网络的预备知识，在第 III-C 节定义插值径向基函数网络，在第 III-D 节讨论插值的误差界，并在第 IV 节展示基准测试。

II. 相关工作

A. 状态格规划器

基于回旋线的状态格运动规划器 [4]自DARPA城市挑战赛 [5] 以来便已应用于自动驾驶车辆，并且在高度无结构且竞争激烈的环境中持续取得成功 [6]。研究也表明，这类规划器很容易集成为可微规划器 [7], [8]。状态格规划器采用的基于采样的方案具有灵活性，便于进行规划代价评估和快速轨迹优化。

B. 数据驱动运动规划

一类研究主张采用完全端到端可训练的自动驾驶技术栈，并直接利用大量数据学习网络策略 [1]–[3]。这类方法具有黑盒算法带来的可扩展性。然而，与更具原则性和传统的运动规划方法相比，完全端到端的方法缺乏可解释性。此外，由于采用黑盒方法，要在这些算法中注入可验证性和安全保证也极为困难。相比之下，另一类研究旨在端到端流水线与传统运动规划之间寻求折中 [8]–[11]。这类方法尝试利用预测和检测模块提供的目标来保留规划的可解释性，并使用一个联合骨干网络，直接接收传感器观测并输出决策，从而利用端到端模型。

C. 可微建模与运动规划

近年来，相关方法更加注重将可微算法应用于自动驾驶。其思路是使现有的基于模型的算法具备可微性。研究表明，可微建模与仿真 [12]–[17]有助于实现更好的控制器综合，并利用强化学习及其他基于学习的方法提高样本效率。可微规划已在通用规划任务 [18]–[21] 以及基于视觉的规划任务和不确定性下的规划 [22]–[24]中展现出良好前景。最后，可微控制 [25]–[28]的性能也已达到传统控制策略的水平，并能提供额外的梯度信息，有助于改进软件技术栈中的上游模块。我们的研究最贴近这一方向，即使基于模型的算法具备可微性。I RBFN 在保留状态格规划器和三次螺旋线的可解释性的同时，采用可微函数逼近器，使其能够以任意精度逼近真实值，从而使整个规划处理流程具备可微性。与类似于 [7], 的方法相比，我们能够获得所有规划器参数的梯度信息，例如，梯度可以从控制或规划损失传递至局部目标选择策略。这使规划器能够真正实现端到端训练，同时不牺牲基于模型的算法的特性。

D. 动力学中的函数逼近器

本文所采用的方法也与 [29], 提出的方法有关，这些方法使用人工神经网络创建可微分的函数逼近，并研究其逼近性质。同样， [30]描述了如何使用径向基函数网络对非线性被控对象模型及其控制器进行表征的方法。然而，这两类方法都没有提供逼近误差界。此外，这些工作的训练数据集并未保持对训练数据的精确拟合。

III. 方法

A. 预备知识

1) 轨迹生成：我们使用回旋线作为阿克曼转向车辆动力学可行轨迹的参数化形式，因为回旋线具有姿态连续性，并且可以表示为多项式。我们将轨迹的曲率表示为弧长 s的三次多项式：

$$\kappa(s) = a + bs + cs^2 + ds^3 \tag{1}$$

根据 [31],，我们重新表述上述三次多项式，使其参数为沿轨迹分布的四个等距点处的曲率：

$$\begin{aligned} a &= \kappa_0 \\ b &= -\frac{1}{2}\frac{-2\kappa_3 + 11\kappa_0 - 18\kappa_1 + 9\kappa_2}{s_f - s_0} \\ c &= \frac{9}{2}\frac{-\kappa_3 + 2\kappa_0 - 5\kappa_1 + 4\kappa_2}{(s_f - s_0)^2} \\ d &= -\frac{9}{2}\frac{-\kappa_3 + \kappa_0 - 3\kappa_1 + 3\kappa_2}{(s_f - s_0)^3} \end{aligned} \tag{2}$$

为了在生成过程中纳入动力学约束，我们采用以下车辆运动学-动力学模型：

$$\begin{aligned} \mathbf{x} = [x, y, \theta, \kappa] \quad \mathbf{u} = \left[v, \dot{\delta}\right] \\ \dot{x}(t) &= v(t)\cos\theta(t) \\ \dot{y}(t) &= v(t)\sin\theta(t) \\ \dot{\theta}(t) &= \kappa(t)v(t) \\ \dot{\kappa}(t) &= \frac{\dot{\delta}(t)}{L} \end{aligned} \tag{3}$$

其中，$\mathbf{x}$ 是状态向量，由车辆的位姿和当前曲率组成；$\mathbf{u}$ 是输入向量，由车辆的速度和转向速度输入组成； L 是车辆的轴距。在转向角较小时，可以用转向角近似车辆的曲率： $\kappa = \tan_L{}^{\delta} \approx \frac{\delta}{L}$。此外，我们还对轨迹的初始状态和终止状态施加约束：—— –

$$\begin{aligned} \mathbf{x}(s_0) &= [x_0, y_0, \theta_0, \kappa_0] \\ \mathbf{x}(s_f) &= [x_g, y_g, \theta_g, \kappa_g] \end{aligned} \tag{4}$$

初始状态约束很简单，因为它由车辆的当前状态决定。终止状态由车辆的目标位姿和目标曲率决定。接下来，我们使用代换 $v = \frac{ds}{dt}$ 重写公式，以求得关于弧长的常微分方程。将等式两边除以 v 后，动力学中的前三项变为以下方程。

$$\begin{aligned} \frac{dx}{ds} &= \cos\theta(s), & x(s) &= \int \cos\theta(s) \\ \frac{dy}{ds} &= \sin\theta(s), & y(s) &= \int \sin\theta(s) \\ \frac{d\theta}{ds} &= \kappa(s), & \theta(s) &= \int \kappa(s)ds \end{aligned} \tag{5}$$

然后将方程 1 用作曲率，执行以下积分，以求得轨迹上的状态。

$$\begin{aligned}\kappa(s) &= a + bs + cs^2 + ds^3 \\ \theta(s) &= as + bs^2 + cs^3 + ds^4 \\ x(s) &= \int_0^{s_f} \cos\left(as + \frac{bs^2}{2} + \frac{cs^3}{3} + \frac{ds^4}{4}\right) ds \\ y(s) &= \int_0^{s_f} \sin\left(as + \frac{bs^2}{2} + \frac{cs^3}{3} + \frac{ds^4}{4}\right) ds\end{aligned} \tag{6}$$

然后，我们可以建立一个优化问题，目标是最小化轨迹的目标位姿（x_g，y_g，θ_g）与最终弧长处的积分位姿之间的欧几里得距离：

$$\text{minimize} \quad |x(s_f)-x_g|^2+|y(s_f)-y_g|^2+|\theta(s_f)-\theta_g|^2 \tag{7}$$

将公式6作为优化的约束条件后，我们可以求得优化变量 $q = [\kappa_0, \kappa_1, \kappa_2, \kappa_3, s_f]$。随后，利用辛普森法则可以高效地计算位置求积梯度和海森矩阵。因此，可以使用牛顿法进行优化。或者，也可以使用鲍威尔法 [32]，在不使用梯度的情况下求得解。为施加动力学约束，我们将公式6中允许的曲率限制在实际物理限制范围内。

2）查找表：利用上一节所述的优化方法，在车辆坐标系中设置局部目标网格，可以创建一个存储优化参数的查找表（LUT）。然而，使用LUT进行高效的在线规划需要较高的网格分辨率。此外，无法利用存储的值准确地对网格点之间的点进行插值。

3）径向基函数网络*:*: 径向基函数网络（RBFN）[33]使用输入到原点的距离的平滑函数，取代S形神经网络中作为神经元的S形函数。我们按照标准定义如下定义径向基函数网络。

定义 III.1（径向基函数网络）。 径向基函数网络由两层组成：包含多个径向基函数神经元的隐藏层和一个线性层。每个径向基函数神经元都以预先定义或可训练的中心为中心，并据此计算距离。

$$\Phi(\mathbf{x}) = \sum_{i=1}^{M} k_i \rho(||\mathbf{x} - c_i||) \tag{8}$$

M 是中心的数量，c_is 是隐藏层径向基函数层的中心，k_i 是线性层的权重。ρ 是选用径向基函数作为径向基函数的平滑激活函数。在我们的应用中，我们使用反二次函数作为核函数：$\rho(\mathbf{z}) = \frac{1}{1+z^2}$。通常，在训练过程中，为了获得更好的近似效果，径向基函数网络的中心选取为训练数据集中的可用数据点。在训练过程中，我们将中心保留为可训练参数。

B. 问题定义

给定类汽车机器人的局部目标 $[x_g, y_g, \theta_g, \kappa_g]$，我们将轨迹生成任务定义为：在工作空间中生成一系列可行控制输入 $\{\mathbf{a}\ \delta\}$，以及相应的位姿和速度曲线 $\{\mathbf{x},\ \mathbf{y}\ \theta,\ \kappa\}$，使机器人从当前位姿到达局部目标。我们进一步构建一个参数化优化问题来描述生成过程。根据第 III-A.1 节，我们可以用多项式参数$[\kappa_0, \kappa_1, \kappa_2, \kappa_3, s_f]$完整地描述单条轨迹。因此，参数化优化问题为：

$$\begin{aligned}\text{minimize} \quad & |x(s_f) - x_g|^2 + |y(s_f) - y_g|^2 + |\theta(s_f) - \theta_g|^2 \\ \text{subject to} \quad & x(s) = \int_0^{s_f} \cos\left(as + \frac{bs^2}{2} + \frac{cs^3}{3} + \frac{ds^4}{4}\right) ds \\ & y(s) = \int_0^{s_f} \sin\left(as + \frac{bs^2}{2} + \frac{cs^3}{3} + \frac{ds^4}{4}\right) ds \\ & \theta(s) = as + bs^2 + cs^3 + ds^4 \\ & a = \kappa_0 \\ & b = -\frac{1}{2}\frac{-2\kappa_3 + 11\kappa_0 - 18\kappa_1 + 9\kappa_2}{s_f - s_0} \\ & c = \frac{9}{2}\frac{-\kappa_3 + 2\kappa_0 - 5\kappa_1 + 4\kappa_2}{(s_f - s_0)^2} \\ & d = -\frac{9}{2}\frac{-\kappa_3 + \kappa_0 - 3\kappa_1 + 3\kappa_2}{(s_f - s_0)^3}\end{aligned} \tag{9}$$

最终，我们将其构建为一个函数逼近问题。我们要逼近函数 f_{opt}，使得：

$$f_{\text{opt}}\left([x_g, y_g, \theta_g, \kappa_g]\right) = \left[\kappa_0^*, \kappa_1^*, \kappa_2^*, \kappa_3^*, s_f^*\right] \tag{10}$$

待逼近的函数以局部目标为输入，并输出描述回旋线的优化参数。下一节将介绍我们提出的函数估计器。

C. 插值径向基函数网络

我们引入了一种改进的径向基函数网络，它在定义域的均匀有限网格上执行插值。结合光滑指示函数，多个径向基函数网络的插值逼近能够产生有界逼近误差。插值径向基函数网络由多个径向基函数网络组成，其中每个径向基函数网络在一个超矩形上执行逼近；该超矩形由定义域上的均匀网格划分而成，如图 2 所示。我们定义了一个光滑且可微的指示函数，该函数针对输入向量的每个维度中的每个区域，返回一个介于 0 和 1 之间的标量。

定义 III.2（光滑指示函数）。 光滑指示函数是定义在超矩形 $r_m = [l_1, u_1] \times \cdots \times [l_N, u_N] \in \mathcal{X}$上的函数 $\gamma_m : \mathbb{R}^N \to [0, 1]$。

$$\begin{aligned}\gamma_n(\mathbf{x}) = \prod_{m=1}^{M} & \left(\frac{\sigma\left(\zeta\left(\mathbf{u}_{m,n} - x_m\right)\right) + 1}{2}\right) \\ & \times \left(\frac{\sigma\left(\zeta\left(x_m - \mathbf{l}_{m,n}\right)\right) + 1}{2}\right)\end{aligned} \tag{11}$$

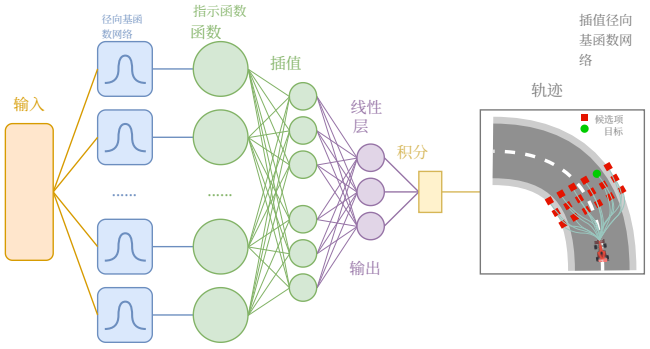

图 2. 轨迹生成用例中 IRBFN 的网络架构。该网络以一组局部目标作为输入，并输出描述期望回旋线的多项式参数。

其中， M 是输入向量的维度， N 是定义的正交多胞体数量， σ 是双曲正切激活函数， ζ 是一个标量参数，而$\mathbf{u}$ 和$\mathbf{l}$ 是定义正交多胞体的区间边界。

区间的边界定义为$\mathbf{s}$的整数倍，其中$\mathbf{s}$是 $\mathcal{X}$ 上划分网格的间距。此外，我们将可控参数 $\delta \in [0,1]$ 定义为 N 和 ζ的函数。通过仔细选择 δ 和ζ的值，可以得到指示函数的一项重要性质。在任意点 $\mathbf{x} \in \mathcal{X}$，$\sum_n \gamma_n(\mathbf{x})$ 的值都可以任意接近 1，从而在所有正交多胞体上实现平滑插值。直观来说，指示函数的设计方式使得在每个维度上，正交多胞体边界处插值径向基函数网络的输出恰好是相邻径向基函数各自输出的一半。插值在网络的最终线性层之前进行。因此，我们将插值径向基函数网络（IRBFN）定义为：

定义 III.3（插值径向基函数网络（IRBFN））。

$$\Phi_{\text{interp}}(\mathbf{x}) = \sum_{i=1}^{N} k_i \rho(||\mathbf{x} - c_i||)\gamma_i(\mathbf{x}) \tag{12}$$

D. 有界插值误差

插值径向基函数网络的一项重要性质是插值误差有界。在下一节中，我们将提供证明，思路如下：首先证明 f_{opt}是可计算函数，因此是连续函数。然后证明，在均匀网格上使用插值径向基函数逼近连续函数时，误差有界。首先，我们根据[34]给出一些重要定义。

定义 III.4（预言机图灵机）。 预言机图灵机（TM）是一种普通图灵机 M，配备了一个额外的查询带以及两个额外的状态：查询状态和回答状态。当机器进入查询状态时，预言机（一个函数 ϕ）会将查询带中的当前字符串 s 替换为字符串 $\phi(s)$，将带头移回查询带的第一个单元格，并使机器 M 进入回答状态。

定义 III.5（k-预言机图灵机）。 A k-预言机 TM M 是一种预言机图灵机，它使用 k 个预言机函数 $\phi_1, \ldots, \phi_k$。 M 可以向第 i 个函数ϕ_i发出查询，$1 \le i \le k$ 方法是在其查询带上写入 $\langle i, n\rangle$，然后第 i 个预言机会在带上写入 $\phi_i(n)$ 进行回答。

定义 III.6（函数的可计算性）。 实函数 $f : [0,1]^k \to R$ 若存在一个 k-预言机 TM M，使得对于所有 $x_1, \ldots, x_k \in [0,1]$ 以及所有$\phi_i \in CF_{x_i}, 1 \le i \le k$，$M^{\phi_1}, \ldots, M^{\phi_k}$都会停机并输出一个二进有理数 d，满足 $|d - f(x_1, \ldots, x_k)| \le 2^{-n}$，则称该函数为可计算的。其中，$CF_{x_i}$ 表示柯西函数，或所有二进制收敛到 x_i的函数集合。

定义 III.7（局部利普希茨连续性）。 若函数 f在有界域 $\mathcal{X}$ 上局部利普希茨连续，则对于$\mathbf{u},\mathbf{v} \in \mathcal{X}$，存在一个有限的 K，使得

$$||f(\mathbf{u}) - f(\mathbf{v})|| \le K||\mathbf{u} - \mathbf{v}|| \tag{13}$$

由于 f_{opt} 在相同初始条件下能在多项式时间内生成梯度下降的唯一解，且解始终存在，因此 f_{opt} 是可计算的。需要指出的是，可计算实函数最早由 Grzegorczyk [35]正式定义。预言机图灵机的定义与原始定义等价。根据 [34]–[36]，可计算函数保持利普希茨连续性。因此，遵循 [37], [38]，我们可以推导出插值径向基函数网络的以下近似界。

定理 III.1（有限均匀网格上的有界插值误差）。 最大插值误差由以下公式统一界定：

$$\begin{aligned} ||\Phi_{\text{interp}}(\mathbf{x}) - f_{\text{opt}}(\mathbf{x})||_\infty < \frac{1}{N^\alpha}\Big[L2^{\frac{\alpha}{2}+1}s^\alpha \\ +2^{\alpha/2}s^\alpha||\Phi_{\text{interp}}||_\infty + ||f_{\text{opt}}||_\infty\Big] \end{aligned} \tag{14}$$

N 是训练样本的数量， α 是 H¨older 阶，L 是 H¨older 常数， s 是训练样本之间的间距，相当于我们定义的网格间距。需要注意的是，在我们的情形中，当函数局部利普希茨连续时， $\alpha = 1$。显然，随着 s 趋于零，这意味着为查找表定义的网格越来越精细，插值误差也趋于零。通过增加样本数量，可以使误差任意小。

E. 可微性

使用可微分的区域指示函数（方程 11），插值径向基函数网络具备完全可微性。在积分步骤中，我们使用 Autograd [39]使梯度能够从采样的局部目标传递到轨迹上的状态。在白盒模型中，轨迹生成流程中的梯度可用性使其在基于模型的强化学习 [17]中相较于黑盒方法具有优势。此外，在为动态系统综合控制器时，加入神经网络组件能够显著提高计算效率 [16]。

表 I 训练数据（查找表）分辨率

维度	值	分辨率
最小值 x(m) 最大值 x(m)	1.0 10.0	0.1
最小值 y (米) 最大值 y (米)	-6.0 6.0	0.1
最小值 θ (弧度) 最大值 θ (弧度)	$-\pi/2$ $\pi/2$	0.1
点数		3213142

IV. 实验

A. 软件实现与训练数据集

轨迹生成流程使用 JAX [40]和 FLAX [41]编写，代码可在线获取：https:// github.com/hzheng40/irbfn。该实现利用即时（JIT）编译来加速数学计算。模型使用自动向量化（vmap）来创建网络的多头结构。最后，在积分步骤中，该实现采用类似 Haskell 的类型签名扫描来消除带有传递变量的 for循环。训练数据集使用牛顿法生成，所用雅可比矩阵和海森矩阵见 [4]。查找表的分辨率在表 I 中指定。由于希望对现有数据进行过拟合，因此所有点都用作训练集。插值径向基函数网络为每个径向基函数网络使用 100 个可训练中心，共包含 880 个区域（径向基函数网络）。定义正交多胞体（区域）的区间长度在 x中为 1.0 米，在 y中为 1.6 米，在 θ中为 0.39 弧度。指示函数中使用的 ζ 在 x中为 15，在 y中为 15，在 θ中为 100。最后，网络使用 Adam [42] 优化器进行训练，学习率为 0.001，采用均方误差损失，批量大小为 2000。在 400 个训练轮次后，整个数据集上的平均训练损失为 0.03107。

B. 基准测试与轨迹生成误差

我们在配备 NVIDIA RTX 2070 Super 图形处理器和 AMD Ryzen 9 3900X 中央处理器的系统上运行所有基准测试。我们对使用不同目标点生成 500 条轨迹的过程进行了分析。采用我们的方法时，显存峰值占用为 273.58 MiB。在 1000 次评估中，每次评估均向目标点添加不同的随机噪声，我们的方法实现了 230.08 赫兹的更新频率。与在线优化多项式解的 3.25 赫兹相比，我们的方法在仅使用少量显存的情况下，速度提升了 70 倍+ 。图3展示了目标设为 $x = 5$ 米时，在不同 y 和 θ 值下的输出示例。

我们还测量了生成轨迹的终点与给定目标点之间的平均误差。在涵盖 x 中 2 至 6 米、 y 中 -4 至 4 米以及 θ 中 -0.3 至 0.3 弧度区域的 500 条轨迹中，我们在表 II 中比较了轨迹终点处的实验误差和理论误差界。理论误差界通过使用公式14获得，然后通过方程 6 中的动力学积分进行传播。我们注意到，尽管 x 和 y 的实验平均误差处于

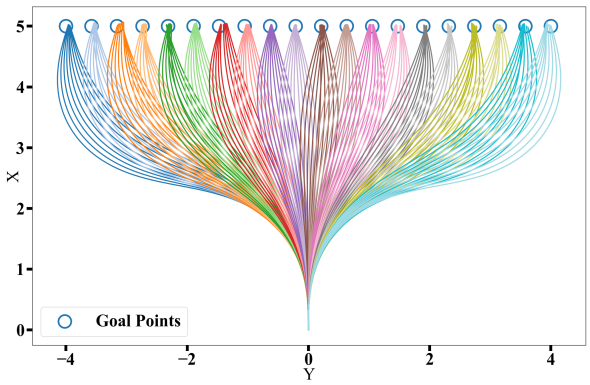

图3. IRBFN 在 $x = 5$ 米处以及不同 y 和 θ 值下生成的轨迹输出示例。

理论误差界内，但 θ 误差高于理论误差界。这可能是由于 θ 维度的内在敏感性所致。因为 θ 的绝对值远小于 x 和 y 的绝对值，而用于训练数据的查找表具有相同的分辨率。此外，由于我们的训练误差不为零，预测误差会叠加到插值误差上，并在积分过程中不断累积，从而生成轨迹上的状态。随着期望目标点向可用训练数据的边缘移动，轨迹生成的准确度还会出现另一个明显的下降。

V. 局限性与结论

局限性： 其中一个局限是，在离线生成查找表时难以筛除无效训练数据。在我们的实验中，我们通过比较弧长以及对应目标的 x 和 y 坐标来剔除无效轨迹。通过提高优化中的迭代次数上限，可以增加有效轨迹的比例。不过，这一步需要在质量和计算时间之间权衡。另一个局限是，本文没有展示利用可微分流程提供的梯度信息来改进规划的潜力。未来可以在端到端流程（即基于模型的强化学习）中结合可训练目标选择策略和 IRBFN。最后，计算梯度信息的计算量明显高于在训练好的网络上进行推理。未来的工作可以对梯度计算进行基准测试并提高其效率。

结论： 本文提出了一种基于插值径向基函数网络的类汽车机器人可微轨迹生成流程。尽管使用 IRBFN 实现完整的规划器栈不在本文讨论范围内，但我们的方法提供了一项重要的理论贡献，证明了其在轨迹生成方面的有效性，并实现了有界的插值误差。在计算效率方面，我们的实现达到了

表 II 轨迹生成端点误差

维度	实验误差	理论误差
x (m)	0.0264	0.0410
y (m)	0.0365	0.0410
θ （弧度）	0.0110	0.0008

与现有方法相比，同时生成500条轨迹时可实现70x+ 的加速，运行频率为 230+ 赫兹，且仅使用少量显存。此外，整个处理流程中每个参数的梯度信息均可获取。

致谢

我们感谢 Matthew O'Kelly 为本项目最初的构思所作的贡献，并感谢 Joshua P. Reddy 参与本项目的初步实验。

参考文献

[1] M. Bojarski、D. Del Testa、D. Dworakowski、B. Firner、B. Flepp、P. Goyal、L. D. Jackel、M. Monfort、U. Muller、J. Zhang等，“用于自动驾驶汽车的端到端学习”，*arXiv*预印本 *arXiv:1604 07316*，2016。.[2] A. E. Sallab、M. Abdou、E. Perot 和 S. Yogamani，“ 自动驾驶的深度强化学习框架 ”，*arXiv* 预印本 *arXiv:1704.02532*，2017。[3] A. Jain、L. Del Pero、H. Grimmett 和 P. Ondruska，“Autonomy 2.0：为什么自动驾驶总是还要等5年？”*arXiv*预印本 *arXiv:2107.08142*，2021。[4] A. Kelly 和 B. Nagy，“通过参数化最优控制生成反应式非完整约束轨迹”，《国际机器人研究期刊》，第22卷，第7-8期，第583–601页，2003。[5] C. Urmson、J. Anhalt、D. Bagnell、C. Baker、R. Bittner、M. Clark、J. Dolan、D. Duggins、T. Galatali、C. Geyer 等，“城市环境中的自动驾驶：Boss与城市挑战赛”，《现场机器人学期刊》，第25卷，第8期，第425–466页，2008，出版商：Wiley Online Library。[6] T. Stahl、A. Wischnewski、J. Betz 和 M. Lienkamp，“动态场景中赛车的多层图轨迹规划”，载于*2019 IEEE*智能交通系统会议（*ITSC*），2019，第3149–3154页。[7] P. Karkus、B. Ivanovic、S. Mannor 和 M. Pavone，“Diffstack：面向自动驾驶车辆的可微模块化控制栈”，*arXiv*预印本 *arXiv:2212.06437*，2022。[8] W. Zeng、W. Luo、S. Suo、A. Sadat、B. Yang、S. Casas 和 R. Urtasun，“端到端可解释神经运动规划器”，载于*IEEE/CVF*计算机视觉与模式识别会议论文集，2019，第8660–8669页。[9] W. Zeng、S. Wang、R. Liao、Y. Chen、B. Yang 和 R. Urtasun，“Dsdnet：深度结构化自动驾驶网络”，载于计算机视觉——*ECCV 2020*，第*28*届欧洲会议，格拉斯哥，英国，*2020*年*8*月，论文集，第*XXI*部分。Springer，2020，第156–172页。[10] A. Cui、S. Casas、A. Sadat、R. Liao 和 R. Urtasun，“Lookout：面向自动驾驶的多样化多未来预测与规划”，载于*IEEE/CVF*国际计算机视觉会议论文集，2021，第16 107–16 116页。[11] S. Casas、A. Sadat 和 R. Urtasun，“Mp3：用于建图、感知、预测与规划的统一模型”，载于*IEEE/CVF*计算机视觉与模式识别会议论文集，2021，第14 403–14 412页。[12] E. Heiden、D. Millard、H. Zhang 和 G. S. Sukhatme，“交互式可微仿真”，*arXiv*预印本 *arXiv:1905.10706*，2019。[13] J. Degrave、M. Hermans、J. Dambre等，“用于机器人学深度学习的可微物理引擎”，《神经机器人学前沿》，第6页，2019。[14] C. Song 和 A. Boularias，“通过可微物理仿真学习推动未知物体”，*arXiv*预印本 *arXiv:2005 05456*，2020。.[15] F. de Avila Belbute-Peres、K. Smith、K. Allen、J. Tenenbaum 和 J. Z. Kolter，“用于学习与控制的端到端可微物理”，神经信息处理系统进展，第31卷，2018。[16] E. Heiden、D. Millard、E. Coumans、Y. Sheng 和 G. S. Sukhatme，“Neuralsim：通过神经网络增强可微仿真器”，载于*2021 IEEE*国际机器人与自动化会议（*ICRA*）。IEEE，2021，第9474–9481页。[17] M. Lutter、J. Silberbauer、J. Watson 和 J. Peters，“面向真实世界离线基于模型的强化学习的可微物理模型”，载于*2021 IEEE*国际机器人与自动化会议（*ICRA*）。IEEE，2021，第4163–4170页。[18] A. Tamar、Y. Wu、G. Thomas、S. Levine 和 P. Abbeel，“值迭代网络”，神经信息处理系统进展，第29卷，2016。[19] N. Nardelli、G. Synnaeve、Z. Lin、P. Kohli、P. H. Torr 和 N. Usunier，“价值传播网络”，arXiv预印本 arXiv:1805.11199，2018。[20] L. Lee、E. Parisotto、D. S. Chaplot、E. Xing 和 R. Salakhutdinov，“门控路径规划网络”，载于国际机器学习大会。PMLR，2018，第2947–2955页。[21] D. S. Chaplot、D. Pathak 和 J. Malik，“使用变换器进行可微空间规划”，载于国际机器学习大会。PMLR，2021，第1484–1495页。[22] P. Karkus、X. Ma、D. Hsu、L. P. Kaelbling、W. S. Lee 和 T. Lozano-P´erez，“用于可组合机器人学习的可微算法网络”，arXiv预印本 arXiv:1905.11602，2019。[23] P. Karkus、D. Hsu 和 W. S. Lee，“QMDP-net：面向部分可观测环境规划的深度学习”，神经信息处理系统进展，第30卷，2017。[24] S. Gupta、J. Davidson、S. Levine、R. Sukthankar 和 J. Malik，“用于视觉导航的认知建图与规划”，载于IEEE计算机视觉与模式识别会议论文集，2017，第2616–2625页。[25] M. Okada、L. Rigazio 和 T. Aoshima，“路径积分网络：端到端可微最优控制”，arXiv预印本 arXiv:1706.09597，2017。[26] B. Amos、I. Jimenez、J. Sacks、B. Boots 和 J. Z. Kolter，“用于端到端规划与控制的可微MPC”，神经信息处理系统进展，第31卷，2018。[27] M. Pereira、D. D. Fan、G. N. An 和 E. Theodorou，“用于序贯决策的受MPC启发的神经网络策略”，arXiv预印本 arXiv:1802.05803，2018。[28] S. East、M. Gallieri、J. Masci、J. Koutnik 和 M. Cannon，“无限时域可微模型预测控制”，arXiv预印本 arXiv:2001.02244，2020。[29] S. Ferrari 和 R. F. Stengel，“ 使用神经网络进行平滑函数逼近”，IEEE 神经网络汇刊，第16卷，第1期，第24–38页，2005。[30] M. Pottmann 和 D. E. Seborg，“一种基于径向基函数模型的非线性预测控制策略”，《计算机与化学工程》，第21卷，第9期，第965–980页，1997，出版商：爱思唯尔。[31] T. M. Howard 等，“面向复杂环境导航的自适应模型预测运动规划”，2009。[32] M. J. Powell，“一种无需计算导数即可求多变量函数最小值的高效方法”，《计算机杂志》，第7卷，第2期，第155–162页，1964。[33] D. S. Broomhead 和 D. Lowe，“径向基函数、多变量函数插值与自适应网络”，皇家信号与雷达研究院马尔文分部（英国），技术报告，1988。[34] K.-I. Ko，《实函数复杂性理论》。Birkhauser Boston Inc.，1991。[35] A. Grzegorczyk，“可计算泛函”，Fundamenta Mathematicae，第42卷，第19553期，第168–202页，1955。[36] ——，“关于可计算实连续函数的定义”，Fund. Math.，第44卷，第61–71页，1957。[37] D. Costarelli，“S形函数逼近及其应用”，数学博士论文，数学与物理系数学分部，罗马大学，2014。[38] A. Bejancu Jr，“有限均匀网格上径向基函数插值的局部准确度”，《逼近理论杂志》，第99卷，第2期，第242–257页，1999。[39] D. Maclaurin、D. Duvenaud 和 R. P. Adams，“Autograd：在NumPy中轻松计算梯度”，载于ICML 2015，第5卷，2015年，第238页，自动机器学习研讨会，第卷第期。[40] J. Bradbury、R. Frostig、P. Hawkins、M. J. Johnson、C. Leary、D. Maclaurin、G. Necula、A. Paszke、J. VanderPlas、S. Wanderman-Milne 和 Q. Zhang，“JAX：Python+NumPy程序的可组合变换”，2018。 [在线]。地址：http://github.com/google/jax [41] J. Heek、A. Levskaya、A. Oliver、M. Ritter、B. Rondepierre、A. Steiner 和 M. van Zee，“Flax：适用于JAX的神经网络库与生态系统”，2023。 [在线]。地址：http://github.com/google/flax [42] D. P. Kingma 和 J. Ba，“Adam：一种随机优化方法”，arXiv预印本 arXiv:1412.6980，2014。