arXiv:2602.08799v1 [cs.RO] 9 Feb 2026

面向网联自动驾驶汽车的通用的面向服务的功能卸载框架

Robin Dehler[1] 和 Michael Buchholz[1]

摘要—**通过将计算任务作为分布式服务分配到本地和远程计算设备，功能卸载是一种有前景的解决方案，可缓解网联自动驾驶汽车（CAVs）或其他自主机器人的计算能力和可用能源方面的限制。本文提出了一种通用的功能卸载框架，可用于卸载任意一组计算任务，并重点关注自动驾驶。为提高灵活性，该功能卸载框架旨在整合不同的卸载决策算法和服务质量（QoS）要求，并可根据不同场景或网联自动驾驶汽车的目标进行调整。着眼于实际应用，我们提出了一种高效的基于位置的方法，根据网联自动驾驶汽车的位置来决定任务是在本地还是远程处理。我们将所提出的框架应用于面向服务的轨迹规划用例，将网联自动驾驶汽车的轨迹规划任务卸载到多接入边缘计算（MEC）服务器。评估通过仿真和真实世界应用两种方式进行。结果表明，该功能卸载框架有望在保障轨迹规划服务质量的同时提升网联自动驾驶汽车的计算效率。此外，仿真结果还表明，该框架能够适应多种场景，包括多个网联自动驾驶汽车同时发起卸载请求的情况。**

索引词—**智能交通系统、分布式机器人系统、联网机器人**

I. 引言

为了实现可靠且安全的应用，自动驾驶系统的许多功能都需要复杂的算法，并对硬件使用率和能耗提出很高要求。由于自动驾驶系统通常具有动态性，例如自动驾驶电动汽车，其可用硬件和电池容量都受到限制。然而，如今网联自动驾驶汽车（CAVs）的功能已不再仅限于车辆内部的硬件和软件，还包括部署在外部计算单元（如服务器）上的硬件和软件组件。

功能卸载是网联自动驾驶汽车利用此类硬件扩展能力的常见方法之一。随着无线通信技术（如车联万物（V2X））的发展，网联自动驾驶汽车能够卸载特定功能，以降低自身的总体工作负载或能耗。可考虑卸载的功能种类繁多，

稿件收到日期：2025年1月10日；录用日期：2025年3月11日。经副编辑及审稿人评议后，编辑 M. Ani Hsieh 推荐本文发表。本研究获得联邦教育与研究部资助（项目 autotech.agil，FKZ 01IS22088W）。
[1]作者为乌尔姆大学测量、控制与微技术研究所成员，地址：德国乌尔姆 D-89081。电子邮箱：{firstname}.{lastname}@uni-ulm.de

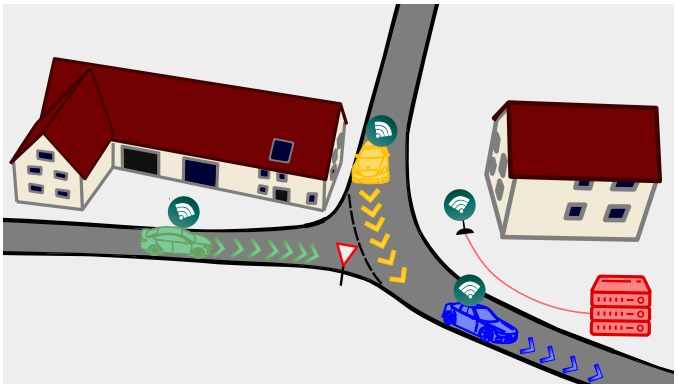

图 1. 所研究的面向服务的功能卸载交叉路口场景，其中包括多辆网联自动驾驶汽车和一台 MEC服务器。

范围涵盖服务质量（QoS）要求较低的信息娱乐和舒适性应用，例如自动空调控制系统 [1]，，也包括服务质量要求较高的安全关键型自动驾驶功能。

针对功能卸载，研究人员开发了多种计算技术，其中两种主要技术是移动云计算（MCC）和多接入边缘计算（MEC）。MCC 是一种利用强大的云服务器作为额外硬件的技术。然而，MCC 的缺点是与自动驾驶系统通信时的时延高度可变，这限制了它在网联自动驾驶汽车中常对时延敏感的功能上的应用 [2]。MEC 通过在网络边缘部署服务器，提供了替代 MCC 的方案，并克服了其局限性，在时延方面具有更好的性能。因此，MEC 是一项重要的功能卸载技术。

面向服务是自动驾驶及一般复杂系统的另一种发展趋势。对于面向服务架构（SOAs），网联自动驾驶汽车（CAV）的每项功能都可以视为一项服务，无需更改整个计算栈即可轻松修改或替换。SOA 的概念前景广阔，因为它能为自动驾驶提供所需的软件灵活性 [3]。功能卸载与 SOA 这两项技术相结合，可为自动驾驶提供一个通用框架。

大量关于功能卸载的研究聚焦于功能卸载决策与资源分配算法，目标是决定哪些功能应从自主设备卸载到服务器。功能卸载决策与资源分配

© 2025 IEEE。允许个人使用本材料。对于任何其他用途，包括在当前或未来的任何媒体中重印/再发布本材料、将本材料用于广告或宣传目的以创作新的合集、转售或再分发至服务器或列表，或在其他作品中重复使用本作品的任何受版权保护的组成部分，必须获得 IEEE 的许可。DOI： 10.1109/LRA.2025.3555939

问题通常被描述为一个优化问题，需考虑例如发射功率、带宽、计算能力、能耗和时延要求 [4]。与文献不同，本工作旨在提供一个功能卸载的总体框架，其中包含卸载任务或功能所需的管理功能。文献中的卸载决策算法例如可以作为一个部分集成到所提出的框架中。功能卸载的其他方面，如可调度性、信息安全和容错分析，不在本文范围内，将在未来工作中考虑。

我们将同一功能在不同计算机上的分布式实现视为不同的服务。根据面向服务的架构（SOA）的理念，借助该框架，这些服务可以轻松适配并互换。所提出的面向服务的功能卸载框架（SOFOF）包含功能卸载的两类实例：为其他方提供服务的服务提供方，以及请求卸载任务的服务请求方。服务提供方和服务请求方的实例相互配合，负责管理多个设备上服务的状态转换。因此，如果该框架未用于现有的 SOA，也可以在汽车面向服务架构（ASOA）的理念下充当服务编排器 [3],否则，相应功能也可以集成到现有编排器中。下文将重点介绍 SOFOF 及其特有功能，并参考例如 [3], 对 ASOA 进行更详细的讨论。使用该框架可以提升网联自动驾驶汽车的计算能力，因为它能够增加可用于自动驾驶任务的服务数量。

为进行评估，我们考虑一个真实世界的郊区交通路口场景，该路口部署了MEC服务器，如图所示。1我们针对面向服务的轨迹规划这一用例，在仿真和真实世界应用中，均于我们的测试场地应用并评估了该框架 [5]。

我们的主要贡献如下：

- 借助SOFOF，我们提出了一个通用框架，功能卸载的两类参与方——服务提供方和服务请求方——均可使用该框架。
- 我们提出了一种高效的、基于位置的功能卸载决策算法，其决策取决于网联自动驾驶汽车的位置，以及专为服务器设定的卸载区域。
- 我们分析了SOFOF在面向服务的轨迹规划用例中的应用。
- 我们通过面向服务的轨迹规划用例，在仿真环境和我们的真实世界自动驾驶测试车辆上对该框架进行了广泛评估，展示了该方法的适用性，尤其是在真实世界场景中的适用性。

II. 相关工作

功能卸载研究主要关注分布式系统中的计算效率优化和资源管理。为解决由此产生的优化问题，大多数方法提出了涉及决策和资源分配算法的解决方案。最具代表性的方法可分为混合整数规划（MIP）方法、李雅普诺夫优化技术以及采用机器学习（ML）的方法 [6], [7]。

当被表述为混合整数规划时，该优化问题通常是NP难的，其规模随远程智能体的数量（例如自动驾驶车辆）而增长 [8]。因此，该问题常采用博弈论求解 [9], [10]。应用博弈论时，车辆作为参与者竞争可用资源，同时考虑其他因素，例如最大时延。李雅普诺夫优化技术的主要优势在于可直接进行稳定性分析，且算法复杂度相对较低 [6]。然而，选择合适的李雅普诺夫函数并非易事。许多机器学习概念将卸载问题描述为马尔可夫决策过程（马尔可夫决策过程），并使用强化学习进行卸载决策 [11], [12], [13], [14], [15]。由于每种概念各有优势，它们常被结合使用，以便扬长避短，例如[16], [17], [18]。需要指出的是，上述方法均在仿真中评估所提出的卸载算法，而未进行真实世界分析。

自动驾驶分布式系统领域的另一个研究重点是不同服务的使用，尤其关注面向服务的架构。研究已证实，面向服务的架构适用于满足网联自动驾驶汽车的需求 [19]。 [3]提出了一种专用于自动驾驶的面向服务架构，称为 ASOA。

由于分布式系统在安全性和信息安全方面面临更大的挑战，因此安全性与信息安全分析是一个研究热点 [20], [19], [21]。 [22], 将车辆协作服务与数字孪生结合，用于协同分布式自动驾驶。 [23]提出了另一种在关键路段进行协同驾驶的方法。 [1],将面向服务的架构扩展至云架构，并分析了具体需求。作者提出了一种编排器设计，用于管理本地服务和云服务的执行。

与 [1], 类似，本文将功能卸载与面向服务的架构相结合，使所提出的框架能够利用不同的分布式服务实现网联自动驾驶。与最新技术水平相比，我们以更全面的视角处理功能卸载任务，将功能卸载决策、资源分配、服务编排和服务质量考量等不同组件整合到一个框架中。面向服务的方法支持不同组件灵活互换，使现有算法（例如用于功能卸载决策的算法）能够轻松集成到该框架中。

III. 功能卸载框架

功能卸载的目标通常通过优化函数来描述，该函数由不同的组成部分构成，例如最小化时延、降低能耗或最大化服务质量。我们提出的面向服务的功能卸载方法尤其关注服务质量，以及额外

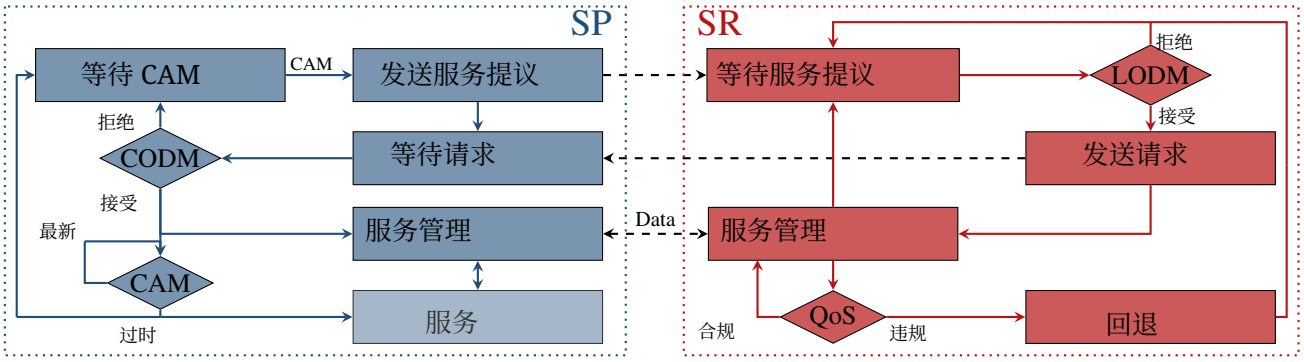

图 2. SOFOF 概览（简化为一个服务提供方和一个服务请求方）。当本地卸载决策（LODM）或集中式卸载决策（CODM）将卸载请求判定为成功后，两个服务管理实例分别处理已卸载服务的数据流。若服务请求方的服务质量要求未得到满足，则服务质量检查会触发回退。服务提供方会持续进行 CAM 检查，以确认网联自动驾驶汽车的位置。

的服务在分布式硬件上相较于仅在本地执行所能带来的改进。为此，我们考虑一个郊区交通场景，其中有多辆网联自动驾驶汽车和一个MEC服务器单元，与图1类似。我们假设网联自动驾驶汽车会持续发送欧洲电信标准协会（ETSI）定义的协作感知消息（CAM） [24]。

SOFOF 包含两个不同的实例：它既可以充当服务提供方（SP），也可以充当服务请求方（SR）。作为服务提供方，SOFOF 会处理传入的服务请求，并为已接受的请求激活和管理服务。作为服务请求方，其目的是分析应卸载哪些任务、请求相应的服务并管理传入的数据。图 2 展示了 SOFOF 在这两种角色下的组成元素。下文将介绍两种不同的运行模式。

A. 服务提供方

如图左侧所示，2，当作为服务提供方运行时，首要任务是等待网联自动驾驶汽车向服务器发送其位置，例如通过协同感知消息发送。为使图示清晰，图中未显示网联自动驾驶汽车中用于持续发送协同感知消息的独立模块。当网联自动驾驶汽车发送的协同感知消息被接收并检测到后，服务提供方会通过服务提供消息向该网联自动驾驶汽车的服务请求方实例告知自身可用，并等待该网联自动驾驶汽车发出请求。收到服务请求方通过特定请求消息发送的请求后，集中式卸载决策算法将决定是否接受或拒绝针对特定服务的请求。我们框架的通用结构使集中式卸载决策算法可以灵活替换。若请求获准，SOFOF 会在单独的线程中执行每项获准的服务，并通过服务管理模块对服务进行管理。图中以较浅颜色显示了该服务请求方所激活服务对应的模块，见图2，因为它并非框架的直接组成部分。

对于某些服务，需要获取来自服务对象服务请求方的额外数据。服务管理通过转换收到的数据，并按照所需格式转发给相应的服务来处理这一需求。服务生成的数据由服务管理单元接收，并通过车联网通信转发给服务请求方。

对于连接到服务提供方的每个服务请求方，都会独立执行所述流程链。不过，具体取决于实现方式，CODM 会了解所有其他正在进行的请求和运行中的卸载服务，因此能够将服务请求方当前的整体容量利用率考虑在内。

B. 服务请求方

作为服务请求方时，SOFOF 首先等待来自 SP 实例的传入服务提供。需要注意的是，服务提供方侧所必需的协同感知消息的持续发送是在一个独立模块中并行完成的；为使图示清晰，该模块未予展示。当收到传入的服务提议时，服务请求方侧的本地卸载决策（LODM）算法会评估所提供服务的收益。与 CODM 类似，LODM 易于替换。如果服务请求方希望使用部分所提供的服务，则会向相应的服务提供方发送一条指明所请求服务的请求消息。

发送请求后，服务管理会等待相应服务的传入数据。当某个已卸载功能的首批数据到达后，该功能将被停用，所接收的数据将用于后续功能，例如车辆控制。

某些服务具有特定的 QoS要求，例如所接收数据的最大时延或最小频率。可为每个已卸载服务配置一个 QoS 控制实例。因此，可根据 QoS要求为每项服务调整不同的 QoS 配置文件。如果某项服务的 QoS 要求被违反，服务管理将激活回退，停止对该特定服务的卸载，并同时在服务请求方上激活该服务，以确保相应功能的服务质量。

IV. 面向服务的轨迹规划

SOFOF 作为一个通用框架，用于处理和管理任意数量的不同功能卸载服务。在本文中，我们特别关注一个用例，即网联自动驾驶汽车的面向服务的轨迹规划。目标是利用功能卸载来提供服务多接入边缘计算服务器上的轨迹规划（MEC-TPL）。服务请求方可以请求并使用 MEC-TPL 服务，作为对

服务本地轨迹规划（CAV-TPL）的补充。在本文中，我们将框架集成到现有的传统自动驾驶架构中。我们仅将轨迹规划功能视为一项服务，所有其他部分均为车辆上的传统功能。在 ASOA 的概念下，我们的框架因此充当多个轨迹规划服务的编排器。

针对我们的使用场景，我们开发了两种用于CODM和LODM的算法，具体如下所述。

A. 集中式卸载决策

部署在SP上的CODM用于接受或拒绝传入的服务请求。在这项工作中，我们采用了一种基于位置的CODM算法，该算法依赖于两个条件。SP存储了地图以及一个以多边形点形式表示的卸载区域，在该区域内可以进行功能卸载。卸载区域可以根据网络连接情况确定。在卸载请求中，SRs将规划路线发送给SP。首先，将接收到的路线与SP可用的地图进行比较。如果存在对应的地图，则使用光线投射算法检查第二个条件，即该路线在可进行卸载的区域内是否足够长 [25]。对于距离，将分析接收到的路线中的路径点，直到第一个路径点位于多边形外部。由于路线信息不包含车辆速度，因此基于SR当前速度采用恒定速度（CV）模型。CV模型作为一种简单的分析方法，用于替代前向规划方法；后者可能会基于CODM算法计算复杂性与性能之间的权衡，考虑道路拓扑和速度限制等因素。给定恒定速度，所用时间取自多边形区域内相邻路径点之间的欧氏距离之和，计算截至多边形内的最后一个路径点v_c，见公式（1）。

对于该条件，使用阈值 t_{min} 来判断卸载是否值得。因此，该条件可表述为

$$t_{min} \leq \sum_{i=1}^{l} \frac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{v_c}, \qquad (1)$$

其中 x_i 和 y_i 分别为航点 i的 x坐标和 y坐标， l 为多边形区域内最后一个航点的索引。

在服务管理处理已激活服务的同时，被服务的网联自动驾驶汽车的最新CAM会持续更新。若该网联自动驾驶汽车离开卸载区域，或最新CAM距今已超过两秒，则终止卸载，并停用已激活的服务。

B. 本地卸载决策

收到的服务提议包含服务提供方位置pos_{SP}的相关信息。采用LODM算法来接受或拒绝该卸载提议。我们再次使用基于位置的决策过程，其概述见算法1。对于LODM，可设置任意卸载半径 r_{off}。该半径决定了服务请求方与相应SP连接点之间允许的最大距离，从而使功能卸载成为可能。

算法 1本地卸载决策

输入： 路径$_{\text{CAV}}$，服务请求方位置 pos_{last}, r_{off}， pos_{SP}, d_{min}
输出： 布尔决策值
1: $d_{\text{passed}} = 0$，位置$_{\text{curr}}$ = 路径$_{\text{CAV}}[1]$
2: **当**位置$_{\text{curr}}$ 与位置$_{\text{SP}}$ < r_{off} 之间的距离**执行**
3: d_{pos} = 位置$_{\text{curr}}$ 与位置$_{\text{last}}$之间的距离
4: $d_{\text{已通过}} \leftarrow d_{\text{已通过}} + d_{\text{pos}}$
5: **如果** $d_{\text{passed}} > d_{\text{min}}$ **则**
6: **返回** 真
7: 位置$_{\text{last}}$ ← 位置$_{\text{curr}}$，位置$_{\text{curr}}$ ← 路径的下一个点$_{\text{CAV}}$
8: **返回**假

算法1 遍历网联自动驾驶汽车的规划路径path_{CAV}，并将各点到SP连接点pos_{SP} 的距离与 r_{off} 进行比较（第2行）。只要该路径位于r_{off}内，便将距离d_{pos} （从当前位置pos_{curr} 到前一位置pos_{pos} ）累加至 d_{passed} （第3-4行）。一旦d_{passed} 大于最小距离 d_{min}，即表示路径path_{CAV} 在 r_{off} 内相对于pos_{SP}已足够长，则接受功能卸载请求（第5-6行）。

C. 轨迹规划

我们使用常见的 ETSI 智能交通系统（ITS）消息 [26]，通过蜂窝网络在网联自动驾驶汽车与 MEC服务器之间双向传输数据。CAV-TPL 和 MEC-TPL 所采用的具体轨迹规划算法已在 [27] 中介绍。简而言之，该算法采用模型预测控制（MPC）方式，使用增广拉格朗日迭代线性二次调节器（AL-ILQR）优化方法来计算轨迹。值得注意的是，不同服务的轨迹规划算法也可以有所不同，例如，为 MEC-TPL 使用更复杂的规划器，以利用 MEC服务器强大的硬件计算更高质量的轨迹，从而为功能卸载带来额外收益。

如前所述，SOFOF的服务管理实例需要管理相应服务的数据流。轨迹规划框架以环境模型和网联自动驾驶汽车的自车运动作为输入。自车运动信息可以通过CAM生成。网联自动驾驶汽车的环境模型通过ETSI ITS集体感知消息（CPM） [28]发送至MEC服务器。MEC服务器上的轨迹规划的另一个优势是，服务器可以直接利用基础设施传感器扩展后的环境模型。MEC-TPL计算出的轨迹由SP实例接收，随后编码为 [29]中提出的自定义机动协调消息（MCM）。由于MCM目前尚未标准化，我们采用了自定义MCM。具体而言，轨迹被编码到 [29] 中MCM的建议机动容器 内，并发送给网联自动驾驶汽车。网联自动驾驶汽车上的服务管理实例接收多媒体消息，提取轨迹，然后将其转发至车辆控制系统。功能激活期间ETSI ITS消息的具体数据流

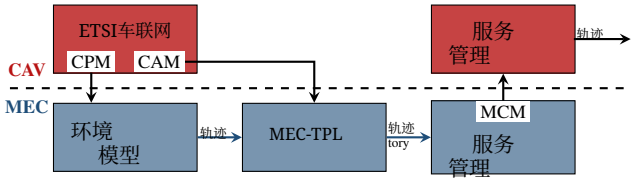

图3. 功能卸载用例——面向服务的轨迹规划中各组件之间的数据流。

面向服务的轨迹规划卸载如图3所示。

D. 服务质量要求

轨迹规划是车辆自动化执行链中的最后几个步骤之一。随后，控制系统会执行车辆控制操作，例如调整转向角或加速度。因此，轨迹规划有严格的服务质量要求。因此，对于面向服务的轨迹规划，我们采用最大时延 $l_{\max}$ 和最大到达间隔时间 $\Delta t_{\max}$ 作为服务质量要求。时延是轨迹生成时间 t_{creation} 与接收到轨迹的时间 $t_{\text{receiving}}$之间的差值。到达间隔时间定义为连续接收到两条轨迹 t_i 和 t_{i+1}之间的时间 $i \in \mathbb{N}_0$。如果这两项要求（汇总为

$$\begin{aligned} l_{\max} &\leq t_{\text{receiving}} - t_{\text{creation}} \\ \Delta t_{\max} &\leq t_{i+1} - t_i \end{aligned} \tag{2}$$

）未得到满足，则会重新激活 CAV-TPL，且 SR 会向 SP 通知服务终止。

V. 评估

我们在仿真和真实世界场景中评估 SOFOF，所考虑的用例是面向服务的轨迹规划。SOFOF 作为 ROS2 节点集成在我们的自动驾驶技术栈中，该技术栈包含用于感知和规划任务的多个节点以及一个车联网模块。动态节点管理通过 ROS2 生命周期节点这一机制实现。

A. 仿真评估

在仿真中，我们比较了 SOFOF 在面向服务的轨迹规划中的应用。

1) 仿真设置： 我们采用文献 [30] 提出的一种基于 ROS2 的软件在环（SIL）仿真框架。SIL 仿真运行于 Ubuntu 22.04 系统，使用 AMD Ryzen 9 7950x CPU 和 64 GB 内存。每辆仿真的网联自动驾驶汽车均分配了自动驾驶技术栈中指定的 ROS2 节点，其中包括 SOFOF 节点。用于仿真 MEC 服务器的节点运行在另一台配备 AMD Ryzen 9 3950X CPU 和 64GB 内存的独立硬件上。两台机器分别位于两个不同的局域网中，并通过光纤连接进行桥接。多跳连接旨在更好地体现

无线连接中的网络状况。卸载消息的发送和车联网通信均通过

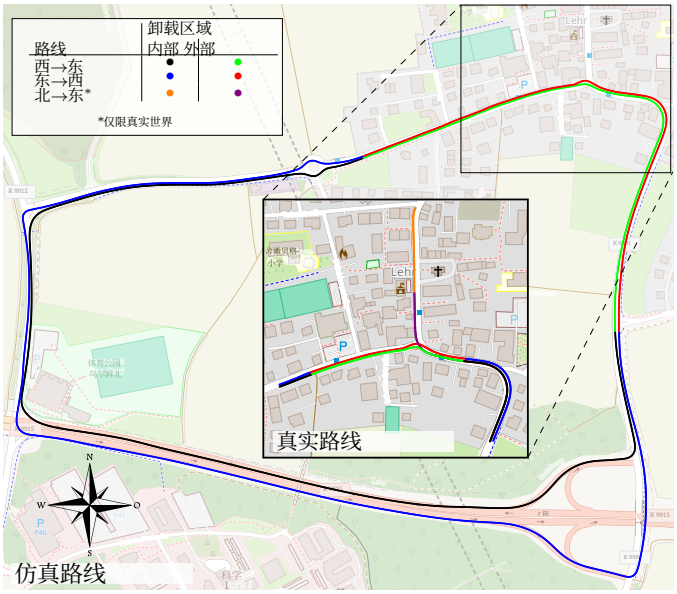

图4. 真实世界街道布局，其中包含试点路口，以及大图中仿真所模拟的两条往返行程。放大的细节图展示了同一路口及真实场景测试的路线。两张图片均使用图例中所示的相同颜色编码标示路线。地图来源： [32]。

模拟 MEC 服务器所在的机器上的 AMQP代理 [31] 进行，并为每辆模拟的网联自动驾驶汽车和 MEC 服务器分别设置客户端。

仿真结果涵盖多个场景，其中网联自动驾驶汽车的数量从1辆到6辆不等。仿真的街道拓扑结构与德国乌尔姆-莱尔的真实世界街道布局相当，其中包含两条往返行程，每条～ 2.9千米。对应的街道布局包含一个试点测试场地，该场地已在 [5]中标出。仿真和真实世界实验的街道布局均如图4所示，背景展示了仿真的往返行程，中间放大的交通路口则展示了真实世界分析所用的路线。颜色变化分别表示不同的卸载区域。可以看出，仿真中的区域相对较大。

我们评估了不同场景，其中改变往返行程中所涉及网联自动驾驶汽车的生成位置和路线，以考虑请求的不同时间条件。对于 CODM，我们选择卸载区域，使每段往返行程中有～ 700m包含在仿真中的 MEC服务器卸载区域内，并为网联自动驾驶汽车的 LODM 设定$r_{\text{off}} = 300\text{m}$ 的卸载半径。在仿真中，网联自动驾驶汽车多次行驶往返行程。卸载区域的大小占往返行程总长度的～ $\frac{1}{4}$ 。

2）服务质量要求：为确定合适的服务质量要求参数，我们首先分析不同参数对不同场景的影响。

图5 展示了在不同场景下，卸载时长 t_{off} 随最大允许到达间隔时间 $\Delta t_{\max}$变化的情况，这些场景涉及不同数量的车辆。 t_{off} 表示轨迹规划服务被卸载的总时长

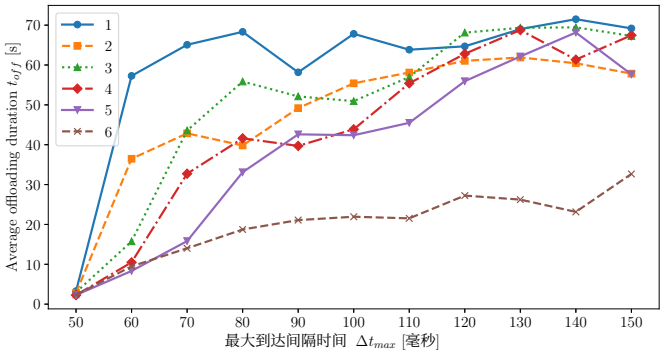

图5. 平均卸载时长 t_{off} 随最大到达间隔时间 $\Delta t_{\max}$ 的变化，场景中网联自动驾驶汽车的数量各不相同。

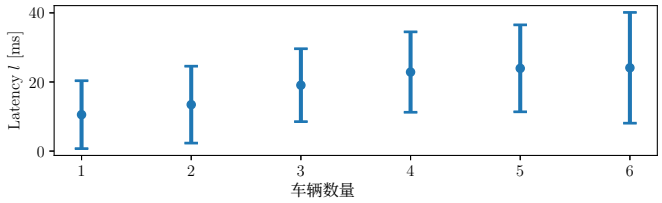

图6. 随仿真车辆数量变化的平均时延值及其标准差。

对于每个已接受的服务请求。我们在不同网联自动驾驶汽车数量 $\in [1,\ldots,6]$ 和 $\Delta t_{\max} \in [10,\ldots,150]$毫秒的条件下，对每种场景运行了 30 分钟，因此总共进行了 33 小时的仿真，其中每辆参与仿真的网联自动驾驶汽车都进行了多次往返行程。随着 $\Delta t_{\max}$增大，在时间限制内可以接收到更多轨迹，这意味着可以在更长时间内进行卸载。图 5也显示了这一点：卸载时长 t_{off} 通常会随着 $\Delta t_{\max}$增大而增加。该场景的最大卸载时长受限于 ~ 70秒，这对应于网联自动驾驶汽车在卸载区域内停留的时间。

图5 还显示了 t_{off}增长较慢，因此车辆越多，卸载时长通常越短。尤其是在有六辆网联自动驾驶汽车的场景中，最大卸载时长低于复杂度较低的场景。首先，更多卸载请求导致MEC服务器的网络拥塞和工作负载增加，进而造成更多服务质量要求违规，因此卸载会更早终止。其次，这一现象与仿真设置有关。当有六辆车时，运行仿真的机器已接近其计算能力限制，这也会缩短这些场景的卸载时长。因此，网络拥塞并非性能下降的唯一原因；更完善的仿真设置或许能改善结果。

对于图5所示的分析，QoS要求中的第二个参数，即最大时延，设为$l_{\max} = 50$毫秒。为评估该数值是否合理，我们记录了仿真中所有接收轨迹的时延。图6展示了平均时延值及其标准差，数值取决于各场景中的仿真车辆

表I 相关系统的平均CPU使用率（以一个 CPU 核心计）。

系统	仿真	真实世界
TPLa c_{TPLa}	19。5%	116.1%
TPLd c_{TPLd}	8.5%	28.3%
SOFOF 服务请求方 $c_{\text{SOFOF SR}}$	0。96%	2.61%

数量。可以看出，平均时延和标准差都会随着车辆数量增加而增大。这分别是车辆增多导致网络拥塞加剧、波动加大的结果。分布还表明，在给定设置下，接收轨迹的时延小于50毫秒的概率很高。

考虑到车辆控制的实时要求，并根据图5中的结果，我们确定了合适的服务质量要求：

$$\begin{aligned} \Delta t_{\max} &= 100\,\text{ms} \\ l_{\max} &= 50\,\text{ms} \end{aligned} \tag{3}$$

3）CPU使用率：功能卸载的主要动机之一是降低电池供电网联自动驾驶汽车的能耗。为此，我们评估了启用和停用功能卸载时，各系统相对于CPU单核容量的平均CPU使用率（百分比），因为所需的计算能力与能耗相关。值得一提的是，由于增加了额外功能，所有网联自动驾驶汽车和MEC服务器的整体CPU使用率显然会更高。不过，我们主要关注的是尽量降低网联自动驾驶汽车的硬件消耗，并假设可用电量有限。

在网联自动驾驶汽车上，未使用和使用SOFOF时的整体CPU使用率分别表示为

$$\begin{aligned} c_{\text{without}} &= t_{\text{total}} \cdot c_{\text{TPLa}} \quad \text{and} \\ c_{\text{with}} &= t_{\text{total}} \cdot c_{\text{SOFOF SR}} + t_a c_{\text{TPLa}} + t_d \cdot c_{\text{TPLd}}, \end{aligned} \tag{4}$$

分别为，其中 t_{total} 表示总时间，t_d表示 t_a TPL停用和启用时的时间。在作为服务请求方时，c_{TPLa}、c_{TPLd}和 $c_{\text{SOFOF SR}}$ 分别表示TPLa、TPLd和SOFOF对应的平均CPU使用率。在另一项仿真中，我们将服务质量要求设定为式（3）所示，并记录了平均CPU使用率，结果见表I的第二列。

以通过功能卸载降低 CPU使用率 为主要目标，可将目标表述为

$$c_{\text{with}} < c_{\text{without}}. \tag{5}$$

将公式（4）代入公式（5），并且 $t_a = t_{\text{tot}} - t_d$时，为实现该目标，卸载应启用的时长占总应用程序运行时间的比例为

$$\frac{t_d}{t_{\text{tot}}} > \frac{c_{\text{SOFOF}}}{c_{\text{TPLa}} - c_{\text{TPLd}}} = 0.0793 \;\widehat{=}\; 7.93\,\%, \tag{6}$$

其中 0。0793 是根据表 I。

在我们的实验中，我们还分析了用于发送协同感知消息、协同感知消息和多媒体消息的 V2X 系统的 CPU使用率

这些消息在网联自动驾驶汽车与多接入边缘计算服务器之间双向传输。然而，假设即使不进行功能卸载，其他 V2X 功能仍会发送协同感知消息，那么功能卸载本身带来的额外使用率可以忽略不计，因此未纳入评估。

可以得出结论，在仿真设置中，如果至少卸载 7.93% 的轨迹规划任务，SOFOF 就能降低网联自动驾驶汽车进行面向服务的轨迹规划时的整体 CPU 工作负载；卸载比例越高，改善效果越显著。

B. 真实世界评估

由于仿真中的时延考量未必能反映该框架在真实世界场景中的应用，我们也在自动驾驶测试车辆和测试场地的多接入边缘计算服务器上部署了 SOFOF [5]。我们以类似仿真的方式评估其对服务质量要求的满足情况，展示该框架在真实世界交通场景动态变化所导致的网络影响下的稳健性。

1）真实世界设置：我们再次分别在测试车辆和多接入边缘计算服务器上使用 ROS2。自动驾驶测试车辆为一辆梅赛德斯-奔驰E级，配备 AMD Ryzen Threadripper 3990X处理器、128GB 内存；多接入边缘计算服务器则配备 AMD Ryzen Threadripper PRO 5965WX处理器、178GB 内存。网络连接通过 5G网络建立，并使用 AMQP 连接处理车联网通信。根据仿真结果，我们将服务质量要求设为 $\Delta t_{\max} = 100\text{ms}$ 和$l_{\max} = 50\text{ms}$。

出于安全考虑，我们为MEC服务器上的CODM定义了与仿真中不同的卸载区域。具体而言，所选卸载区域涵盖了我们试点交通路口附近预定义路线中的 $\sim 220\text{m}$，见图4。实际可用区域可能大得多。我们在图 15 所示区域内的三条不同路线进行了测试驾驶，见图4。在此过程中，SOFOF算法会根据自动驾驶测试车辆的位置和所定义的卸载区域，动态地将轨迹规划服务从CAV-TPL切换为MEC-TPL。与仿真分析类似，我们监测了相关ROS2节点所需的CPU使用率，即轨迹规划（TPLa和TPLd）和功能卸载（SOFOF SR）；由于SOFOF SP运行在由有线电源供电的MEC服务器上，因此未将其纳入监测。

2）*CPU* 使用率：表I 的第三列显示了自动驾驶测试车辆各个组件观测到的平均 CPU 使用率。比较仿真和真实世界分析的两列数据可以发现，对于单辆车，TPLa 在真实世界评估中的 CPU 使用率显著高于仿真中的使用率。这是由于 TPLa 的参数设置有所不同：真实世界应用中，MPC规划算法的更新频率（见 [27],）设为100Hz，而非仿真中的20 Hz。较高的更新频率能确保更安全的轨迹规划，因为它可以更快地适应车辆周围环境的变化。值得注意的是，服务质量要求没有调整，因为如果无法满足时延和到达间隔时间要求，所选值仍可作为回退方案。此外，由于我们以单个 CPU 核心为基准计算百分比，因此该值可能达到 > 100 %，这意味着该进程需要多个 CPU 核心才能运行。

在真实世界测试车辆上实现SOFOF，也比仿真中需要占用更高比例的CPU。其原因与TPLa数值较高的原因类似。由于SOFOF需要处理已发布的轨迹，因此发布频率越高，CPU使用率也越高。

根据公式（5）中的目标，要使 SOFOF 系统使用较少的 CPU，轨迹规划必须停用的时间比例为

$$\frac{t_d}{t_{\text{tot}}} > 0.0615 \;\widehat{=}\; 6.15\,\%. \tag{7}$$

我们还记录了从 MEC服务器发送到测试车辆的每条轨迹的时延。对于真实世界场景，均值和标准差为

$$\begin{aligned} l_{\text{avg}}^{\text{real}} &= 11.94\,\text{ms} \\ l_{\text{std}}^{\text{real}} &= 2.98\,\text{ms}. \end{aligned} \tag{8}$$

尤其是，公式（8）中的标准差会随图 6 6 $l_{\text{avg}}^{\text{sim 1}} = 10.54$ 所示 $l^{\text{sim 1}} = 9.83$ 的某个仿真场景中的数值而变化。具体而言，对于一辆车，图中显示 ,ms 和 ,std ms。标准差较低的一个原因是，在真实世界分析中，只有与功能卸载相关的数据会发送到服务器，而在仿真中，还需要将额外的仿真数据发送到模拟服务器的机器。

根据观测值 $l_{\text{avg}}^{\text{real}}$, $l_{\text{std}}^{\text{real}}$和 $l_{\max} = 50\,\text{ms}$，极高比例的轨迹都在最大延迟内收到。然而，在总共 2 个15 个已进行的场景中，由于在图 $l_{\max}$ 所示的卸载区域内发生了违规，功能卸载不得不提前终止。4。然而，违规导致的动态切换回 CAV-TPL 并未影响乘客的驾驶体验，这表明即使网络条件不断变化，各项轨迹规划服务之间仍能实现平稳、安全的切换。

因此，可以说，真实世界实验不仅支持仿真结果，还表明应用 SOFOF 能够有效扩展我们自动驾驶测试车辆的本地计算硬件和软件。

六、结论与未来工作

本文介绍了 SOFOF，这是一种面向服务架构（SOA）下的通用功能卸载框架，可扩展自动驾驶任务可用的服务数量。本文以轨迹规划卸载任务为例，将其作为一个用例引入，并通过该框架进行处理。评估结果通过仿真展示了 SOFOF 在提升网联自动驾驶汽车硬件使用率方面的优势。真实世界评估表明，该框架适用于实际场景，也显示出功能卸载对未来移动出行系统的发展具有巨大潜力。

由于采用通用设计，SOFOF 为广泛的后续研究开辟了空间。未来工作将着重改进系统，通过引入多种不同服务（包括协作功能，例如协作式交叉口管理），进一步发挥功能卸载的优势 [33]。我们还将着重完善仿真设置，以更好地评估 SOFOF 的可扩展性，并考虑安全性：除了采用 QoS 要求外，还将通过分析不同服务的下游数据，评估卸载任务的质量。

参考文献

[1] M. Sommer、H. Guissouma、M. Schindewolf 和 E. Sax，“面向云端汽车电子电气架构动态扩展的编排器”，载于 面向服务的计算，M. Aiello、J. Barzen、S. Dustdar 和 F. Leymann 编，Cham：Springer Nature Switzerland，2025 年，第 24–41 页。[2] H. Wang、T. Liu、B. Kim、C.-W. Lin、S. Shiraishi、J. Xie 和 Z. Han，“基于云/边缘/雾计算的网联车辆架构设计方案”，*IEEECommunications Surveys& Tutorials*，第 22 卷，第 4 期，第 2349–2377 页，2020 年。[3] A. Kampmann、B. Alrifaee、M. Kohout、A. Wüstenberg、T. Woopen、M. Nolte、L. Eckstein 和 S. Kowalewski，“面向高度自动化车辆的动态面向服务软件架构”，载于*2019 IEEE*智能交通系统会议（*ITSC*），2019 年，第 2101–2108 页。[4] J. Du、L. Zhao、J. Feng 和 X. Chu，“具有最小最大公平性保证的混合雾/云计算系统中的计算卸载与资源分配”，*IEEE Transactions on Communications*，第 66 卷，第 4 期，第 1594–1608 页，2018 年。[5] M. Buchholz、J. Müller、M. Herrmann、J. Strohbeck、B. Völz、M. Maier、J. Paczia、O. Stein、H. Rehborn 和 R.-W. Henn，“利用基础设施传感器基于多接入边缘计算服务器环境模型处理自动驾驶中的遮挡问题”，*IEEE Intel- ligent Transportation Systems Magazine*，第 14 卷，120，2022 3，第 106 no. pp. – 。[6] F. Saeik 、 M. Avgeris 、 D. Spatharakis 、 N. Santi 、 D. Dechouniotis、J. Vio- los、A. Leivadeas、N. Athanasopoulos、N. Mitton 和 S. Papavassiliou，“边缘和云计算中的任务卸载：数学、人工智能与控制理论解决方案综述”，*Computer Networks*，第 195 卷，文章 108177，2021 年。[7] A. Acheampong、Y. Zhang、X. Xu 和 D. Kumah，“边缘计算系统中当前任务卸载算法、策略与方法综述”，*Computer Modeling in Engineering & Sciences*，第 134 卷，第 1–54 页，2022 年 8 月。[8] K. Zhang、Y. Mao、S. Leng、Q. Zhao、L. Li、X. Peng、L. Pan、S. Maharjan 和 Y. Zhang，“5G异构网络中的节能卸载”，*IEEE Access*，第 4 卷，第 5896–5907 页，2016 年。[9] J. Zhao、Q. Li、Y. Gong 和 K. Zhang，“车联网中云辅助移动边缘计算的计算卸载与资源分配”，*IEEE Transactions on Vehicular Technology*，第 68 卷，第 8 期，第 7944–7956 页，2019 年。[10] Z. Ning、P. Dong、X. Wang、X. Hu、J. Liu、L. Guo、B. Hu、R. Y. K. Kwok 和 V. C. M. Leung，“5G赋能车联网中的部分计算卸载与自适应任务调度”，*IEEE Transactions on Mobile Computing*，第 21 卷，第 4 期，第 1319–1333 页，2022 年。[11] N. Zhao、Z. Ye、Y. Pei、Y.-C. Liang 和 D. Niyato，“无人机辅助移动边缘计算中的多智能体深度强化学习任务卸载”，*IEEE Transactions on Wireless Communications*，第 21 卷，第 9 期，第 6949–6960 页，2022 年。[12] J. Wang、J. Hu、G. Min、A. Y. Zomaya 和 N. Georgalas，“基于元强化学习的边缘计算快速自适应任务卸载”，*IEEE Transactions on Parallel and Distributed Systems*，第 32 卷，第 1 期，第 242–253 页，2021 年。[13] Q. Luo、T. H. Luan、W. Shi 和 P. Fan，“基于深度强化学习的多无人机协同目标搜索计算卸载与轨迹规划”，*IEEE Journal on Selected Areas in Communications*，第 41 卷，第 2 期，第 504–520 页，2023 年。[14] H. Li、K. D. R. Assis、S. Yan 和 D. Simeonidou，“面向多服务器边缘计算网络任务卸载策略的基于深度强化学习的长期资源规划”，*IEEETransactionsonNetwork andService Management*，第 19 卷，第 4 期，第 4151–4164 页，2022 年。

[15] L. Huang、S. Bi 和 Y.-J. A. Zhang，“无线供能移动边缘计算网络中的在线计算卸载深度强化学习”，*IEEE Transactions on Mobile Computing*，第 19 卷，第 11 期，第 2581–2593 页，2020 年。[16] W. Sun、H. Zhang、R. Wang 和 Y. Zhang，“降低 6G 数字孪生边缘网络中的卸载时延”，*IEEE Transactions on Vehicular Technology*，第 69 卷，第 10 期，第 12 240–12 251 页，2020 年。[17] S. Bi、L. Huang、H. Wang 和 Y.-J. A. Zhang，“李雅普诺夫引导的深度强化学习：移动边缘计算网络中的稳定在线计算卸载”，*IEEE Transactions on Wireless Communications*，第 20 卷，第 11 期，第 7519–7537 页，2021 年。[18] Y. Dai、K. Zhang、S. Maharjan 和 Y. Zhang，“数字孪生网络中的随机计算卸载深度强化学习”，*IEEE Transactions on Industrial Informatics*，第 17 卷，第 7 期，第 4968–4977 页，2021 年。[19] S. Kugele、P. Obergfell、M. Broy、O. Creighton、M. Traub 和 W. Hopfensitz，“关于汽车软件的面向服务”，载于*2017 IEEE*软件架构国际会议（*ICSA*），2017 年，第 193–202 页。[20] M. Rumez、D. Grimm、R. Kriesten 和 E. Sax，“汽车面向服务架构概述及其对安全对策的影响”，*IEEE Access*，第 8 卷，第 221 852–221 870 页，2020 年。[21] H. Zhu、R. Lu、X. Shen 和 X. Lin，“面向服务的车载网络中的信息安全”，*IEEE Wireless Communications*，第 16 卷，第 4 期，第 16–22 页，2009 年。[22] Y. Hui、X. Ma、Z. Su、N. Cheng、Z. Yin、T. H. Luan 和 Y. Chen，“协作即服务：数字孪生赋能的协同式分布式自动驾驶”，*IEEE Internet of Things Journal*，第 9 卷，第 19 期，第 18 607–18 619 页，2022 年。[23] K. Zhang、A. Yang、H. Su、A. de La Fortelle、K. Miao 和 Y. Yao，“连续静态关键路段异构无人驾驶车辆的面向服务协作模型与机制”，*IEEE Transactions on Intelligent Transportation Systems*，第 18 卷，第 7 期，第 1867–1881 页，2017 年。[24] ETSI EN 302 637-2，“智能交通系统（ITS）；车载通信；基本应用集；第 2 部分：协同感知基本服务规范”，2014 年 9 月，最终草案 V1.3.1。[25] S. D. Roth，“用于实体建模的光线投射”，*Computer Graphics and Image Processing*，第 18 卷，第 2 期，第 109–144 页，1982 年。[26] ETSI ES 202 663，“智能交通系统（ITS）；在 5 GHz频段运行的智能交通系统物理层与媒体访问控制层的欧洲配置文件标准”，2009 年 11 月，最终草案 V1.1.0。[27] J. Ruof、M. B. Mertens、M. Buchholz 和 K. Dietmayer，“利用动态优化面向城市环境的实时空间轨迹规划”，载于*2023 IEEE*智能车辆研讨会（*IV*），2023 年，第 1–7 页。[28] ETSI TR 103 562，“智能交通系统（ITS）；车载通信；基本应用集；集体感知服务（CPS）分析；第 2 版”，2019 年 12 月，v2.1.1。[29] M. B. Mertens 、 J. Müller 、 R. Dehler 、 M. Klimke、M. Maier、S. Gherekhloo、B. Völz、R.-W. Henn 和 M. Buchholz，“支持城市场景与混合交通的扩展机动协调协议”，载于*2021 IEEE*车载网络会议（*VNC*），2021 年，第 32–35 页。[30] J. Strohbeck、J. Müller、A. Holzbock 和 M. Buchholz，“DeepSIL：利用多个轨迹预测网络评估运动规划方案的软件在环框架”，载于*2021 IEEE/RSJ*智能机器人与系统国际会议（*IROS*），2021 年，第 7075–7081 页。[31] OASIS，*OASIS*高级消息队列协议（*AMQP*）*1.0* 第 *3* 部分：消息传递，2012 年 10 月 29 日，OASIS 标准。 [在线]。可访问：http://docs.oasis-open.org/amqp/core/v1.0/ [32] OpenStreetMap 贡献者，“来自 https://planet.osm.org 的 Planet 转储文件”，https://www.openstreetmap.org/#map=16/48.42884/9.96415，2025 年。[33] M. B. Mertens、J. Müller 和 M. Buchholz，“利用概率预测进行无信号交叉口混合交通协同机动规划”，载于*2022 IEEE*智能车辆研讨会（*IV*），2022 年，第 1174–1180 页。