arXiv:2603.01864v1 [cs.CV] 2 Mar 2026

使用端点感知建模的实时流式轨迹预测

Alexander Prutsch David Schinagl Horst Possegger 格拉茨技术
大学视觉计算研究所{alexander.prutsch, david.schinagl, possegger}
@tugraz.at

摘要

邻近交通参与体的未来轨迹对自动驾驶车辆的路径规划与决策具有重要影响。尽管轨迹预测是一个研究充分的领域，但现有研究主要聚焦于基于快照的预测，即将每个场景独立于其全局时间上下文进行处理。然而，真实世界的自动驾驶系统需要在连续设置下运行，要求对数据流进行低延迟的实时处理，并在连续时间步上给出一致的预测。我们利用这一连续设置，提出一种轻量且高精度的基于流式处理的轨迹预测方法。我们通过一种新颖的端点感知建模方案，整合先前预测中的有价值信息。我们的时间上下文传播将先前预测结果的轨迹端点作为锚点，以提取有针对性的场景上下文编码。该方法能够高效引导场景编码器提取高度相关的上下文信息，而无需细化迭代或分段解码。实验表明，我们的方法能够有效地在连续时间步之间传递信息。与采用多阶段细化处理的方法不同，我们的方法显著降低了推理延迟，因而非常适合真实场景部署。我们在 *Argoverse 2* 多智能体与单智能体基准测试上取得了最先进的流式轨迹预测结果，同时所需资源大幅减少。项目页面。

1. 引言

轨迹预测，即预测附近交通参与者未来的运动，是自动驾驶车辆自主系统中的关键组成部分。轨迹预测对于实现高效、可靠的导航至关重要，因为它能够让自动驾驶车辆预判周围智能体的行为，并根据预测的运动主动规划机动动作。轨迹预测模型的延迟直接影响从传感器观测到车辆控制动作之间的总延迟，因为轨迹预测通常在感知与运动规划模块之间执行。因此，要在自动驾驶车辆中实际部署，轨迹预测模型必须以极低的延迟生成高度准确的预测。

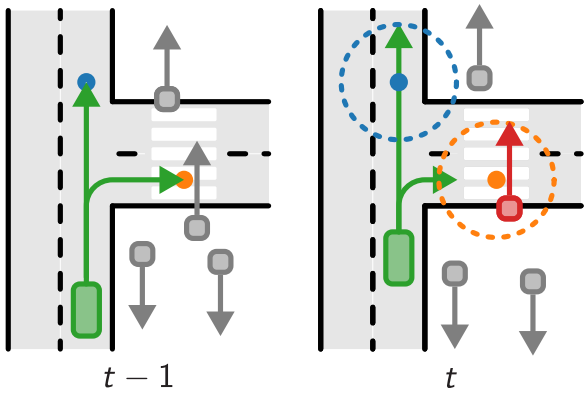

图 1。交通场景不断变化，而在快照上运行、而非处于连续设置中的轨迹预测方法往往忽略了这一点。例如，在预测焦点智能体（绿色）的转弯轨迹时，前一次估计的端点附近（时间 $t-1$）与行人（红色）可能发生的交互，是当前预测（t）需要考虑的重要上下文。我们通过一种新颖的端点流式机制对这种时间上下文传播进行建模，从而以极低的延迟实现准确预测。

轨迹预测的一大挑战是识别对运动预测影响最大的地图元素和最相关的邻近智能体，尤其是在预测时域较长（通常超过 100 米）的复杂场景中。尽管智能体的初始位置为相关上下文元素提供了强有力的先验，但未来路径上的场景元素也发挥着至关重要的作用，如图 1所示。近期的轨迹预测方法通常采用细化方案 [28, 42, 43] 来利用这些信息：对初始的粗略未来轨迹进行迭代细化，以提高预测准确性。例如，SmartRefine [42]提出了一种自适应多阶段细化方案，并通过重新编码场景上下文来改进强基线模型 QCNet 的预测 [43]。采用此类细化迭代的方法在 Argoverse 2（AV2）等大规模基准上取得了很高的准确性 [37]。然而，这些方法会带来较高的计算成本和延迟。

这限制了它们在现实世界自动驾驶系统中的适用性。在这些系统中，快速高效的决策至关重要，而延迟较高的方法可能会因传感器观测过时而导致结果可靠性降低。

此外，当前轨迹预测研究着重于处理和评估基准数据集中单个场景的快照 [9, 37]。这种基于快照的处理方式与真实自动驾驶车辆的应用形成鲜明对比，在实际应用中，轨迹预测系统运行于连续环境中。在这种现实场景下，必须考虑额外的约束。尤其是，连续预测之间的一致性对于集成到下游决策系统至关重要。另一方面，连续的流式处理也有其优势，例如利用多个预测帧中的上下文信息来提高未来运动估计的准确性。近期，开创性的流式处理方法 [24, 30]相继涌现，为应对连续轨迹预测的挑战奠定了基础。RealMotion [30] 表明，简单的骨干架构结合流式机制，在 AV2 数据集上取得了良好的结果 [37]。然而，他们的方法并未利用所有可用的时序信息来编码场景上下文，例如没有考虑过去的预测。

为推动连续、稳定且高效的真实世界运动预测，我们提出了一种新颖的基于流式处理的轨迹预测方法，利用时间信息有针对性地提取场景上下文。具体而言，我们提出了流式端点感知建模（SEAM）架构，以有效整合连续帧中的信息。通过将上一预测帧的预测端点用作先验信息，我们无需进行耗费资源的细化迭代，即可将目标区域信息直接融入模型。我们在 AV2 [37]，单智能体和多智能体基准测试上的深入评估表明，我们提出的新型流式机制相较于此前提出的流式技术具有显著优势 [30]。我们证明，该方法优于更复杂、资源需求更高的模型，同时还解决了真实世界中的实际问题，例如减少同一智能体多次预测之间的波动。该方法兼具高准确性和极低的推理延迟，在 AV2 基准测试的基于流式处理的轨迹预测任务中取得了最先进成果。总之，我们的主要贡献包括：

- 我们提出了一种新的轨迹预测模型，能够有效利用场景的连续性进行以目标为中心的上下文编码，无需任何细化步骤。
- 尽管推理延迟极低，我们仍证明了模型在复杂轨迹预测任务中表现出色，并在 AV2 多智能体基准测试的流式处理方面创下新的最先进水平。
- 消融研究凸显了端点感知建模在从前一个预测时间步传播信息方面的有效性。

2. 相关工作

轨迹预测方法通常以智能体状态和地图信息作为输入。地图数据主要建模为向量化数据 [12, 32] ，并通过类似 PointNet 的架构 [26], 进行编码；智能体时间序列信息则通常由循环架构 [21, 32], 、基于注意力的模块 [4, 17, 25, 30] 或状态空间模型 [39]处理。为了捕捉智能体与地图元素之间的关系，近期方法采用图神经网络 [6, 12, 16, 18, 31, 38] 或基于注意力的编码器 [4, 11, 19, 22, 23, 28, 41]。轨迹解码通过多层感知机（MLP） [4, 30]或更先进的基于交叉注意力的设计 [28, 29, 39]完成。

2.1. 轨迹预测的细化模块

迭代细化是最先进的轨迹预测方法中的常见策略 [11, 28, 34, 42, 43]。Argoverse 2 [37] 和 WOMD [9] 等长距离基准涉及超过 100 米的预测，因此会产生大量场景上下文元素。虽然智能体的初始位置有助于识别相关上下文，但仅依据给定的智能体过去运动，要检测远距离的关键元素仍然具有挑战性。为了解决这一问题，可以应用多个细化步骤，逐步提取对智能体未来行为有重大影响的场景上下文元素 [28, 42]。初始预测为识别关键场景元素提供了有力指导，并可利用这些元素迭代细化轨迹假设。

MTR [28] 及其后续工作 [7, 11, 29] 通过使用动态意图查询在解码过程中聚合场景上下文，迭代细化轨迹预测。R-Pred [5]提出了一种两阶段预测方法：初始预测定义场景上下文管道，随后细化网络利用这些管道生成改进后的预测。QC-Net [43] 采用两阶段解码流程，在细化阶段将初始预测作为先验，融入交叉注意力解码器的模式查询中。此外，在每个推理步骤中，该方法将未来轨迹预测划分为较短的时间片段，并通过循环方式进行预测。SmartRefine [42]提出了一种自适应细化迭代策略，并在 QCNet 上进行了验证。在每个推理步骤中，该方法根据更新后的场景编码，自适应地执行一定次数的轨迹解码迭代，逐步细化预测。最近，DeMo [39] 引入了一种解耦查询策略，先分别建模运动状态一致性和多模态意图，再解码最终的轨迹输出。

总体而言，DeMo [39] 和SmartRefine [42]等方法有力地证明了多阶段解码能够提升

轨迹预测质量。然而，准确性的提升往往很有限，尤其是考虑到在每个预测时刻执行多阶段细化方案所需的计算成本显著更高。这种收益递减凸显了在设计用于现实世界应用的轨迹预测模型时，需要采用新的策略来平衡计算成本与准确性。

2.2. 流式轨迹预测

当前的轨迹预测研究主要聚焦于基于快照的预测，即在处理单个场景时，不考虑不断演变的场景中的全局时间上下文。然而在实践中，自主系统运行于连续环境中，并以流式方式处理数据，这不仅带来了时间一致性等额外要求，也提供了利用更多时间信息来改进预测的机会。

Pang 等人 [24]较早地探索了连续流式场景中的运动预测。他们提出的预测流处理器通过自定义遮挡推理模块和可微滤波器扩展了基于快照的轨迹预测模型，以增强时间一致性。Real-Motion [30] 表明，即使采用简单的骨干网络，结合来自前一帧的上下文和预测也能获得准确结果 [4]。他们提出了一种上下文流，在当前帧和先前帧的自运动对齐上下文上应用交叉注意力；还提出了一种轨迹中继机制，通过嵌入的过去预测所产生的偏移量来优化当前预测。DeMo [39]也在其方法中整合了这两种机制。HPNet [31] 在相对简单的 AV1 数据集上，探索了基础的历史预测注意力模块 [3]。QCNet [43]探索了用于过去观测的键值缓存，涉及与流式处理相关的概念。然而，它既未考虑过去的预测，也未确保各预测步骤之间的对齐；此外，其计算密集型解码器也给实时部署带来了重大挑战。

总的来说，现有的基于流式处理的方法采用相应机制来保持连续预测 [24, 30]的一致性，并利用以智能体为中心的场景上下文在各时间步 [30] 之间的传播，以应对现实世界自动驾驶的连续性。然而，这些方法并未探索将先前预测的轨迹用作上下文编码先验知识的做法。鉴于这些区域对于提取相关上下文特征 [28, 42], 的重要性，我们提出了一种高效策略，在单个解码步骤中纳入这些信息，从而避免细化模块带来的高计算开销。

3. 使用端点感知建模的流式轨迹预测

我们提出 SEAM，这是一种基于流式处理、兼具高准确性与轻量化的轨迹预测方法范式。与基于细化的方法不同 [28, 42],，后者通过多阶段解码聚合相关场景上下文，迭代细化预测；SEAM 则引入了一种新机制，在各个时间步之间传递预测，以提取以端点为中心的目标区域特征。这一设计避免了细化迭代带来的计算开销，使 SEAM 能够以低延迟实现高准确性。此外，通过利用过去的预测来识别潜在目标区域，SEAM 解决了以往基于流式处理的方法的一项关键局限 [24, 30],：它们忽视了这一强大的时间先验。我们的方法与现实世界中自动驾驶的连续特性高度契合：随着未来预测时间步的推进，与其他智能体的交互可能发生变化，因此必须动态地加以考虑。例如，请回想图1：在预测转弯时，必须考虑人行道上可能横穿的行人。如概览图2所示，我们利用过去预测的端点引导模型关注相关区域，从而只需一个解码步骤即可实现高度准确的未来轨迹预测，无需计算开销巨大的细化迭代。

我们方法的核心组件是基于流式处理的双区域编码机制，该机制利用过去预测的端点来增强场景上下文聚合。除了以焦点智能体当前位置为中心的传统以智能体为中心的上下文编码外，我们还引入了第二组上下文特征：通过提取先前预测轨迹端点附近的场景元素，获得以目标为中心的特征，因为这些区域很可能与当前帧的预测高度相关。我们以轨迹端点作为坐标系原点，使用浅层目标编码器对其进行编码。在解码过程中，我们同时对以智能体为中心和以目标为中心的特征执行注意力操作。

我们在第 3.1 节和 3.2 节介绍基线编码。接下来，我们介绍端点感知建模（*EAM*），其中包括新颖的以目标为中心的特征编码（第 3.3 节），以及将其整合到所提出的双上下文解码器中（第 3.4 节）。第 3.5 节概述了基线流式机制的整合 [30]。我们在第3.6节介绍了向多智能体场景的扩展。

输入表示：轨迹预测是指根据历史智能体状态A 和车道数据 L来预测未来轨迹 F。我们遵循最先进研究 [4, 25, 30, 39] 中常用的输入表示，首先对焦点智能体周围固定半径范围内的所有智能体和地图元素进行采样。张量 $A \in \mathbb{R}^{N_a \times T_h \times D_a}$ 表示历史轨迹，其中 N_a 是智能体数量， T_h 表示历史时间步的数量， D_a 表示运动状态的特征维度，即位置和速度。每条车道沿其中心线采样P_l 个点来表示。我们将车道数据定义为张量 $L \in \mathbb{R}^{N_l \times P_l \times D_l}$，其中 N_l 表示车道段的数量

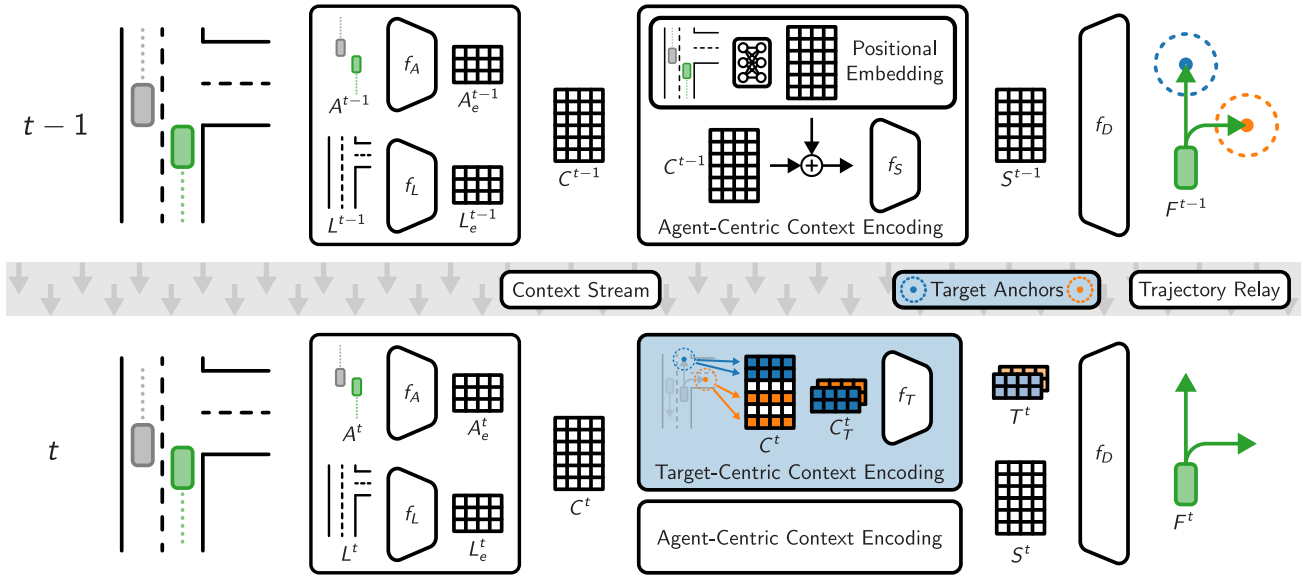

图 2. 我们提出的基于流式处理的轨迹预测架构 SEAM 概览。我们假设在帧 $t-1$ 中，焦点智能体尚无此前的预测。因此，只执行一次标准的轨迹预测模型轮次。在下一帧 t中，我们将此前预测的端点纳入其中，以汇总以目标为中心的上下文信息。我们使用第二条编码器路径对其进行编码，并将其作为额外输入提供给我们新颖的双上下文解码器。

以及 D_l 表示特征维度，即车道中心线的xy坐标。为便于高效编码历史运动和车道数据，我们首先将每个元素转换到局部坐标系中。我们根据初始位姿对历史轨迹进行归一化，而车道中心线的坐标则以其中心位置为基准。这使模型能够独立于全局位置学习运动模式和车道段形状。

输出表示： 我们的模型为焦点智能体预测多模态未来轨迹假设$F \in \mathbb{R}^{K\times T_f\times 2}$，其中 T_f 表示未来时间步的数量，K 表示运动模式的数量，同时还预测相应的概率分数 $P \in \mathbb{R}^K$。我们使用 xy坐标表示未来轨迹。因此，轨迹预测可表述为 $(F, P) = f(A, L)$，其中f 是我们的轨迹预测模型。更具体地说，我们的模型由一个编码器组成，该编码器生成场景上下文 $S = f_E(A, L)$；以及一个轨迹解码器 f_D，它利用场景上下文 $S \in \mathbb{R}^{N_a+N_l\times D}$ 来预测轨迹 F和概率分数 P，即 $(F, P) = f_D(S)$。

流式处理： 为适应现实世界中的自动驾驶应用，我们采用流式处理方案，在当前时间步 t预测未来轨迹。我们利用先前编码的上下文信息 S^{t-1} 和过去的轨迹预测 F^{t-1} 生成高精度预测，并确保轨迹在时间上的一致性。因此，我们在时间t 的轨迹预测可表述为 $(F^t,\ P^t) = f\ (A^t,\ L^t,\ S^{t-1},\ F^{t-1})$ 。

3.1. 智能体与车道编码

我们遵循 [17, 25]，通过使用线性层将智能体数据 A^t 投影到 D维特征空间 $\mathbb{R}^{N_a\times T_h\times D}$，对智能体历史运动进行编码。接下来，我们沿时间维度应用多头自注意力 [33]，以捕捉智能体运动动力学，并使用池化操作聚合时间信息。由此得到智能体表示 $A_e^t \in \mathbb{R}^{N_a\times D}$，其中每个智能体的运动都编码在一个特征向量 $\mathbb{R}^{1\times D}$中。对于车道数据的编码，我们遵循默认的最先进方法 [4, 25, 28–30, 39]，使用一个类似迷你 PointNet 的 [26]网络。这将得到车道上下文表示 $L_e^t \in \mathbb{R}^{N_l\times D}$。因此，我们的编码器 f_A 和 f_L 生成用于进一步编码的智能体特征令牌 $A_e^t = f_A\ (A^t)$ 和车道特征令牌$L_e^t = f_L\ (L^t)$ 。

3.2. 以智能体为中心的上下文编码

分别对每个场景上下文元素进行编码后，必须学习所有场景元素之间的关系。为此，我们将智能体 A_e^t 和车道 L_e^t 词元拼接起来，形成场景上下文表示 $C^t \in \mathbb{R}^{N_c\times D}$，其中 $N_c = N_a + N_l$。为纳入类别信息（例如不同的智能体和车道类型），我们将可学习类型嵌入 [4] 添加到场景标记中。编码地图元素与智能体之间关系的一种常见方法是在场景标记之间应用自注意力 [4, 17, 25, 39, 42]。为此，必须建立全局位置编码，通常使用以焦点智能体为中心的坐标系来实现。我们使用每个局部标记坐标系的位置（x，y）和旋转角度 yaw 对全局位姿进行编码，

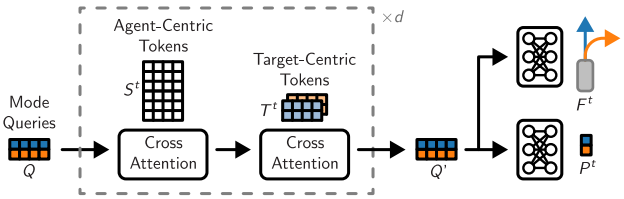

图 3. 我们新颖的双上下文注意力解码器架构，同时利用以智能体为中心的 S^t 和以目标为中心的 T^t 特征。

即。智能体轨迹起点和车道段中心。我们对偏航角应用正弦和余弦变换，并使用两层多层感知机（MLP）处理位姿信息 $\mathbb{R}^{N_c\times 4}$。随后，我们将位置嵌入添加到上下文令牌 $C^t \in \mathbb{R}^{N_c\times D}$ 中，并在场景编码器 f_S中应用多头注意力。由此得到编码后的场景上下文 $S^t = f_S(C^t)$，其中包含 $S^t \in \mathbb{R}^{N_c\times D}$。

3.3. 以目标为中心的上下文编码

以智能体为中心的编码会隐式引导模型关注焦点智能体周围的区域，因为附近的上下文元素与未来运动预测密切相关。然而，正如图 1所示，智能体即将前往的区域对于准确预测也至关重要。由于我们采用基于流式处理的设置，前一帧中可见的每个智能体都有多模态历史轨迹预测，记作 $F^{t-1} \in \mathbb{R}^{K\times T_f\times 2}$。这些先前的预测 F^{t-1} 为场景区域提供了强有力的先验信息，而这些区域对于上下文提取同样至关重要。我们使用这些轨迹的端点作为锚点，来定义 K 目标区域。接下来，我们从场景上下文 C^t 中收集每个目标锚点周围感兴趣区域半径 r 内的词元，形成 K 独立的目标上下文令牌集$C_T^t \in \mathbb{R}^{K\times N_c'\times D}$。实际上，$N_c'$ 远小于 N_c 因此，与主要的以智能体为中心的编码相比，我们的目标编码计算开销更低。

与处理以智能体为中心的场景上下文 S 的流程相同，我们在目标编码器 f_T 中分别对每个以目标为中心的上下文 C_T^t进行编码，得到 $T^t \in \mathbb{R}^{K\times N_c'\times D}$。为捕捉全局坐标原点与各目标锚点坐标系之间的关系，我们还为每个区域向目标锚点的变换添加位置嵌入（详情见补充材料）。

3.4. 双上下文注意力解码

我们的解码器 f_D 基于以智能体为中心的场景上下文S^t 生成未来轨迹 F^t 和概率分数 P^t 以及我们新颖的以目标为中心的场景上下文 T^t，其形式化表示为$(F^t,\ P^t) = f_D(Q,\ S^t,\ T^t)$。我们采用可学习查询$Q \in \mathbb{R}^{K\times D}$ 来获得多模态输出，并实现类似 DETR 的 [2] 解码方法。图3展示了我们的双上下文注意力解码器的架构。

我们迭代执行 d次交叉注意力，作用于场景上下文 $S \in \mathbb{R}^{N_c\times D}$和目标专用令牌 $T \in \mathbb{R}^{K\times N_c'\times D}$。因此，在每个解码器阶段都能利用来自两个特征集的信息，并将其整合到最终输出中。以智能体为中心的上下文 S 提供局部信息，而目标上下文 T 则为模型提供关于与其他智能体及地图元素之间长时域依赖关系的细粒度信息。每个模式查询仅关注一组与之关联的目标词元 $T \in \mathbb{R}^{1\times N_c'\times D}$。最后，我们使用一个专用于每种输出的双层多层感知机，将解码后的查询 Q' 投影为未来轨迹 $F^t \in \mathbb{R}^{K\times T_f\times 2}$和概率分数 $P^t \in \mathbb{R}^K$。

3.5. 上下文引用与轨迹中继

除了新颖的端点感知建模之外，我们还利用上下文引用和轨迹中继 [30]，通过从前一帧 $t-1$ 传递信息来建模流式设置。为确保公平比较，我们对这两种机制均采用 [30] 的设置：为考虑焦点智能体在 $t-1$ 与 t 之间的运动，我们在将场景上下文 S^{t-1} 通过智能体到场景及地图到地图的交叉注意力整合到当前编码 C^t 之前，对其应用运动感知层归一化（MLN） [35] 。轨迹中继采用基于交叉注意力的交互模块，利用过去的预测 F^{t-1} 来细化当前预测 F^t。

3.6. 扩展到多智能体设置

为了准确预测场景中所有智能体 N_a 的轨迹，我们首先为每个智能体单独生成边缘预测。为此，我们将每个智能体依次作为焦点智能体输入我们的单智能体模型，堆叠输入数据，并以 N_a的批量大小执行一次推理。接下来，我们引入一个轻量级全局一致性模块，将边缘预测融合为联合的、全局一致的世界预测。

我们的一致性模块以每个智能体的模态特征 $Q' \in \mathbb{R}^{N_a\times K\times D}$ 及其全局位置$R \in \mathbb{R}^{N_a\times 4}$ 作为输入。一个浅层多层感知机将全局位置编码为位置嵌入。随后，我们对每个智能体的所有模态（在 K上）以及对每个模态的所有智能体（在 N_a上）应用自注意力，以捕捉智能体内部关系与智能体间关系。遵循我们的单智能体解码器设计，我们使用两个浅层多层感知机解码最终的世界轨迹预测。作为输出，我们得到 $F_w \in \mathbb{R}^{K\times N_a\times T_f\times 2}$ 个世界预测及其相应的置信度分数 $P_w \in \mathbb{R}^K$。

4. 实验

我们在 Argoverse 2（AV2） [37]数据集的单智能体和多智能体基准上展示了实验结果，并对这两种设置进行了延迟分析，以评估我们方法的实际适用性。此外，我们还针对 AV2 和 Argoverse（AV1） [3], 上的端点感知建模进行了消融研究，更多结果见补充材料。

方法	流式处理	minADE_1	minFDE_1	MR_6	minADE_6	minFDE_6	brier-minFDE_6
SIMPL [40]	✗	2.03	5.50	0.19	0.72	1.43	2.05
HPTR [41]	✗	1.84	4.61	0.19	0.73	1.43	2.03
Forecast-MAE [4]	✗	1.74	4.36	0.17	0.71	1.39	2.03
MTR [28]	✗	1.74	4.39	0.15	0.73	1.44	1.98
EMP-D [25]	✗	1.75	4.35	0.17	0.71	1.37	1.98
GANet [34]	✗	1.77	4.48	0.17	0.72	1.34	1.96
ProIn [8]	✗	1.77	4.50	0.18	0.73	1.35	1.93
RealMotion [30] *	✓	1.65	4.12	0.15	0.67	1.29	1.93
QCNet [43]	✗	1.69	4.30	0.16	0.65	1.29	1.91
RealMotion [30] (训练/验证)*	✓	<u>1.59</u>	<u>3.93</u>	0.15	0.66	1.24	1.89
Tamba [14]	✗	1.66	4.24	0.17	0.64	1.24	1.89
ProphNet [36]	✗	1.80	4.74	0.18	0.68	1.33	1.88
MTR++ [29]	✗	1.64	4.12	<u>0.14</u>	0.71	1.37	1.88
DyMap [10]	✗	-	-	-	0.71	1.29	1.87
SmartRefine [42]	✗	1.65	4.17	0.15	<u>0.63</u>	<u>1.23</u>	<u>1.86</u>
DeMo [39]	✓	**1.49**	**3.74**	**0.13**	**0.61**	**1.17**	**1.84**
SEAM（我们的方法）	✓	1.60	3.95	0.15	0.66	1.24	**1.84**

表1. Argoverse 2 测试集上的单智能体轨迹预测结果。对于所有指标，数值越低越好；表格按 brier-minFDE 降序排列$_6$。我们报告了官方排行榜上已发表作品的结果，且未采用模型集成。Pang 等人的 [24] 流式处理方法未纳入比较（其自定义数据集导致无法直接比较）。*对于 RealMotion [30]，我们报告了其官方检查点的结果，以及其论文中的结果；后者使用训练集和验证集进行训练。

4.1. 实验设置

数据集： 我们在 Argoverse 2（AV2） [37] 轨迹预测数据集上评估模型。该数据集的采样频率为 10 Hz，包含 199,908 个训练场景、24,988 个验证场景和 24,984 个测试场景，采集于六座美国城市。每个场景的前 5 秒作为历史上下文，目标是预测随后 6 秒的轨迹。较长的历史上下文使 AV2 非常适合用于流式处理中进行评估，因为它支持合理的滑动窗口处理（见下文）。相比之下，WOMD [9]仅提供 1 秒的历史数据，而 nuScenes [1] 提供 2 秒历史数据，但采样频率只有 2 Hz。将本就有限的历史数据点进一步拆分到多个窗口中，会限制智能体编码器学习运动模式的能力。相关工作 [24, 30, 39]也未将这两个数据集用于流式处理。我们在 AV2 的单智能体基准（预测每个场景中一个智能体的轨迹）和多智能体基准（预测每个场景中多个智能体的全局一致轨迹）上进行评估。此外，我们还在 Argoverse 1（AV1） [3] 上开展消融研究；该数据集包含 2 秒历史数据，并考察未来 3 秒的情况。

指标： 我们使用标准的 AV2 [37] 基准指标评估我们的方法。对于单智能体情形，我们使用漏检率（MR_k）、最小平均位移误差（minADE_k）、最小最终位移误差（minFDE_k）和 Brier 最小最终位移误差（brier-minFDE_k）。按照官方排行榜的做法，我们使用排名前 $k \in \{1, 6\}$ 的评分轨迹进行评估。对于多智能体情形，我们采用 actorMR_k、avgMinADE_k、avgMinFDE_k 和 avgBrierMinFDE_k，这些指标用于计算每个世界中所有行动体的平均误差。

流式处理： 为了实现流式处理在现有基准上进行评估，我们遵循 [30, 39]，将历史模型输入 T_h 限制为可用历史数据的一部分。随后，我们采用滑动窗口并执行多次模型推理。具体而言，对于 AV2，我们使用时长为 $h = 3$s 的历史输入，并在 $t \in \{3, 4, 5\}$s 时进行三次预测。最终预测与基于快照的方法在同一时间点进行，从而确保比较公平。我们在补充材料中详细比较了流式处理与基于快照的处理。

4.2. Argoverse 2 单智能体结果

表1 展示了 Argoverse 2 单智能体测试集上的结果。我们的 SEAM 与 DeMo [39], 在主要指标（brier-minFDE_6）上取得了最佳分数，优于 Tamba [14] 等近期工作，甚至胜过 QCNet [43] 和 SmartRefine [42]等计算资源开销更大的方法。无需任何细化迭代便取得如此出色的表现，彰显了端点感知建模的有效性。我们也优于 RealMotion [30], 这一先前提出的基于流式处理的预测方法，同时实现了更低的推理延迟（第 4.4节）。DeMo [39]的位移误差略低，这是因为其解耦解码能够生成更多样的轨迹，但延迟显著更高（第4.4节）。minFDE_6 与 brier-minFDE_6 之间的差距很小（主要指标的概率惩罚所致），这表明我们的模型具有很强的置信度估计能力，这对于将模型集成到规划模块也至关重要。这说明，直接集成以目标为中心的特征有助于评估机动可能性。

方法	流式处理	avgMinADE$_1$	avgMinFDE$_1$	行动体MR$_6$	avgMinADE$_6$	平均最小终点位移误差$_6$	平均 Brier 最小终点位移误差$_6$
FJMP [27]	✗	1.52	4.00	0.23	0.81	1.89	2.59
Forecast-MAE [4]	✗	1.30	3.33	0.19	0.69	1.55	2.24
RealMotion [30]	✓	1.14	2.97	0.18	0.62	1.32	2.01
DeMo [39]	✓	1.12	2.78	0.16	0.58	1.24	1.93
SEAM（我们的方法）	✓	1.05	2.57	0.16	0.62	1.21	1.85
QCNeXt [44]	✗	**1.03**	**2.55**	**0.14**	**0.54**	**1.13**	**1.79**

表 2. Argoverse 2 测试集上的多智能体轨迹预测结果。所有指标的值越低表示性能越好，表格按 avgBrierMinFDE$_6$ 降序排列。所有方法的结果均未采用模型集成。

方法	流式处理	延迟（$B=1$）		延迟 ($B=32$)		延迟 ($B=64$)		模型参数	AV2 测试集 brier-minFDE$_6$
		离线	在线	离线	在线	离线	在线		
SmartRefine [42]	✗	>118 毫秒*		>680 毫秒*		-		8.0M	1.86
QCNet [43]	✗	118 毫秒		680 毫秒		-		7.7M	1.91
RealMotion [30]	✓	**86 毫秒**	**24 毫秒**	247 毫秒	88 毫秒	436 毫秒	150 毫秒	**2.9M**	1.93
DeMo [39]	✓	139 毫秒	39 毫秒	275 毫秒	83 毫秒	466 毫秒	142 毫秒	5.9M	**1.84**
SEAM（我们的方法）	✓	91 毫秒	28 毫秒	**140 毫秒**	**50 毫秒**	**246 毫秒**	**74 毫秒**	4.6M	**1.84**

表 3. 使用 NVIDIA V100 GPU 在单智能体设置下预测 B Argoverse 2 场景的延迟分析。在线（针对基于流式处理的方法）是处理单个滑动窗口所需的时间；离线 是处理完整数据流（以生成基准预测）所需的时间。对于流式方法，在线 延迟与实际应用相关。*SmartRefine 未提供 AV2 的实现——由于其主要结果是在 QCNet 上增加细化迭代，因此延迟高于 QCNet。

4.3. Argoverse 2 多智能体结果

以往的方法，例如， [14, 42], 通常只报告单智能体结果。然而，多智能体基准与实际系统更为相关：在现实环境中部署的模型必须能够同时预测一个场景中多个智能体的轨迹。表2中的结果表明，在这种设置下，我们的 SEAM 在主要指标上优于先前基于流式处理的方法 [30, 39] 。目前表现最佳的方法是快照方法 QCNeXt [44], , 它是 QCNet 的扩展 [43]。然而，两者的位移误差仅相差几厘米，而 QCNeXt 在解码时大量使用递归和细化迭代，对实时部署构成了重大挑战。由于模型延迟会直接影响传感器观测与预测输出之间的延时，计算开销较大的模型在现实场景中可能因响应延迟而导致更差的结果。作为参考，以 50 km/h 行驶的智能体在 50 ms 内约行驶 0.7 m。因此，在处理延迟增加的情况下，位移误差仅略有降低，在实际应用中并无益处。

4.4. 延迟分析

Argoverse 2 单智能体设置： 表 3 比较了在 AV2 单智能体场景中预测不同批次大小时的延迟。与相关工作相比，我们的方法运行时间更具优势，尤其是在较大的批次大小下扩展性良好；这在实践中尤为重要，因为单智能体基准测试每个场景只评估一个智能体。SEAM 仅使用单个解码器遍，因此延迟最低，不同于 DeMo [39], QCNet [43] 和 SmartRefine [42], , 它们采用多个解码阶段。此外，我们无需像 QCNet 和 SmartRefine 那样将完整的未来轨迹拆分为多个片段，而是直接进行预测。与 RealMotion 相比， [30] 我们更高效的解码器设计带来了延迟优势。快速的处理速度使我们的方法非常适合真实场景部署，因为它将流式建模与计算效率结合起来——这两点是实际应用中的关键考量。

Argoverse 2 多智能体场景： 关于基于流式的处理的相关工作（RealMotion [30] 和 DeMo [39]）未提供其多智能体实现的详细信息。此外，QCNeXt 没有官方实现 [44]。因此，无法直接比较延迟。不过，根据任务性质，我们可以合理地假设，多智能体场景下的每场景延迟高于表 3中报告的单智能体延迟。对于我们的模型，在 AV2 验证集上使用单张 NVIDIA V100 GPU（批量大小 $B=1$）时，我们测得多智能体场景下的平均在线延迟为 38 ms/场景。值得注意的是，这甚至快于 DeMo 和 QCNet 报告的单智能体预测延迟，凸显了我们方法的高效性。

4.5. 消融研究

信息流机制： 在表 4 中，我们将端点感知建模（EAM）（包括以目标为中心的特征编码和双上下文解码器）与先前工作 [30]提出的上下文流（CS）和轨迹中继（TR）流式机制进行比较。每组的第一行展示了结果

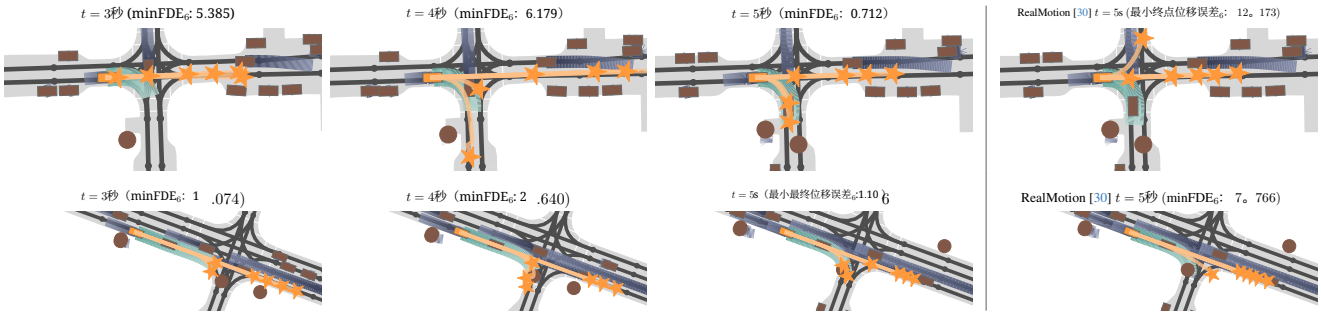

图 4. Argoverse 2 中两个场景的定性结果。我们展示了 基于流式处理的方法在 $t \in \{3, 4, 5\}$s 时的可视化结果。可视化还展示了**未来真实值**、**智能体历史轨迹**和**邻近智能体**。右列展示了 RealMotion 在基于流式处理的设置下于 $t = 5$s 时的最终预测 [30]。上排：焦点智能体正接近一个交叉路口，此时其他交通流正在经过，因此难以判断可能的行驶方向。下排：一名行人在右侧交叉路口穿过街道。我们的方法正确预测出车辆可以直行或右转：既可以在人行横道前等待，也可以在人行横道上的行人已经离开后直接通行。

数据集	h	流	CS+TR [30]	EAM	mADE$_6$	mFDE$_6$	b-mFDE$_6$
AV1 Val	2s	✗	✗	✗	0.64	1.04	1.67
	1s	✓	✓	✗	0.62	0.98	1.62
	1s	✓	✗	✓	0.60	0.95	1.55
	1s	✓	✓	✓	**0.59**	**0.93**	**1.52**
AV2 Val	5s	✗	✗	✗	0.77	1.39	2.00
	3s	✓	✓	✗	0.75	1.29	1.93
	3s	✓	✗	✓	0.70	1.27	1.88
	3s	✓	✓	✓	**0.66**	**1.25**	**1.85**

表 4。流式轨迹预测中不同信息转发机制的比较。我们将 [30] 提出的上下文流（CS）和轨迹中继（TR）集成方式与我们的端点感知建模（EAM）进行比较。第一行展示了传统的基于快照的使用方式。我们突出显示了**最佳**值。

我们的骨干架构在传统的基于快照的预测设置下运行，且不使用任何流式机制。将使用CS和TR的基于流式的处理与我们的方法进行比较，凸显了我们的EAM在从前一帧传播信息方面的有效性。我们的预测端点感知建模所取得的提升大于 [30]提出的两种流式机制。在流式设置中利用端点信息被证明对提高轨迹预测准确性非常有益。总体而言，将我们的EAM与CS和TR相结合可获得最佳结果。

目标上下文半径： 我们评估了不同目标上下文半径 r 在表 5中的影响。半径达到 30 米时，性能趋于饱和，因为在更大的半径下纳入更多上下文，并不会为我们的端点感知建模提供相关信息。更大的半径会增加目标上下文令牌的数量，从而提高延迟，因此在实际应用中，30 米的半径是性能与效率之间的最佳权衡。值得注意的是，即使半径较小，为 10 米，我们的方法仍然有效，因为将先前的端点位置注入解码器已提供了强有力的先验。在解码过程中纳入端点位置，也能引导交叉注意力关注

数据集	r	mADE$_6$	mFDE$_6$	b-mFDE$_6$
AV2 Val	10m	0.68	1.27	1.89
	15m	0.67	1.26	1.87
	30m	**0.66**	**1.25**	**1.85**
	45m	**0.66**	**1.25**	**1.85**

表5. 不同半径 r 的比较，用于为我们的以目标为中心的区域选择上下文元素。我们突出显示**最佳**值。

以智能体为中心的特征，进一步凸显了端点信息在流式处理中的重要性。

4.6. 定性结果

图 4展示了 AV2 验证集中的两个场景的预测结果，说明了流式处理在 $t \in \{3, 4, 5\}$秒内不断演进的特性。为展示与以往流式轨迹预测研究相比，我们的端点感知建模所带来的优势，我们还展示了 RealMotion [30]在 $t = 5$秒时的输出。更多定性结果（包括失败案例）见补充材料。

5. 结论

我们提出了一种用于流式轨迹预测的新型端点感知建模方法。我们的双上下文方法能够有效融入持续演变的相关场景上下文，无需进行代价高昂的轨迹细化迭代，即可获得高精度预测。对于实际场景规模，我们的推理速度显著快于高度迭代的方法 [29, 42] 和以往的流式方法 [30, 39]，同时在 Argoverse 2 单智能体数据集上取得了具有竞争力的结果，并在多智能体基准上为流式轨迹预测树立了新的最先进水平。

致谢。 本研究部分由 Addsafety (923936)、由 BMIMI 资助的 aCOMET 项目、BMWET 以及施蒂利亚州共同资助。COMET 计划由 FFG 管理。

参考文献

[1] Holger Caesar、Varun Bankiti、Alex H Lang、Sourabh Vora、Venice Erin Liong、Qiang Xu、Anush Krishnan、Yu Pan、Gian- carlo Baldan 和 Oscar Beijbom。nuScenes：面向自动驾驶的多模态数据集。发表于 CVPR，2020年。6[2] Nicolas Carion、Francisco Massa、Gabriel Synnaeve、Nicolas Usunier、Alexander Kirillov 和 Sergey Zagoruyko。使用 Transformer 进行端到端目标检测。发表于 ECCV，2020年。5[3] Ming-Fang Chang、John Lambert、Patsorn Sangkloy、Jagjeet Singh、Slawomir Bak、Andrew Hartnett、De Wang、Peter Carr、Simon Lucey、Deva Ramanan 等。Argoverse：借助富地图进行三维跟踪和预测。发表于 CVPR，2019年。3, 5, 6, 11[4] Jie Cheng、Xiaodong Mei 和 Ming Liu。Forecast-MAE：利用掩码自编码器进行运动预测的自监督预训练。发表于 ICCV，2023年。2, 3, 4, 6, 7, 11, 12[5] Sehwan Choi、Jungho Kim、Junyong Yun 和 Jun Won Choi。R-Pred：基于管查询注意力的两阶段运动预测与轨迹细化。发表于 ICCV，2023年。2[6] Alexander Cui、Sergio Casas、Kelvin Wong、Simon Suo 和 Raquel Urtasun。GoRela：通过相对运动实现视点不变运动预测。发表于 ICRA，2023年。2[7] Tobias Demmler、Lennart Hartung、Andreas Tamke、Thao Dang、Alexander Hegai、Karsten Haug 和 Lars Mikelsons。用于运动Transformer轨迹预测的动态意图查询。发表于 IEEE IV，2025年。2[8] Yinke Dong、Haifeng Yuan、Hongkun Liu、Wei Jing、Fangzhen Li、Hongmin Liu 和 Bin Fan。ProIn：学习基于渐进式交互进行轨迹预测，以实现自动驾驶。arXiv，2024年。6[9] Scott Ettinger、Shuyang Cheng、Benjamin Caine、Chenxi Liu、Hang Zhao、Sabeek Pradhan、Yuning Chai、Ben Sapp、Charles R Qi、Yin Zhou 等。面向自动驾驶的大规模交互式运动预测：Waymo开放运动数据集。发表于 ICCV，2021年。2, 6[10] Bin Fan、Haifeng Yuan、Yinke Dong、Zhengyu Zhu 和 Hongmin Liu。利用地图相关任务进行双向智能体-地图交互特征学习，以实现自动驾驶轨迹预测。《IEEE自动化科学与工程汇刊》，2025年。6[11] Yiqian Gan、Hao Xiao、Yizhe Zhao、Ethan Zhang、Zhe Huang、Xin Ye 和 Lingting Ge。MGTR：利用激光雷达进行运动预测的多粒度Transformer。发表于 ICRA，2024年。2, 11[12] Jiyang Gao、Chen Sun、Hang Zhao、Yi Shen、Dragomir Anguelov、Congcong Li 和 Cordelia Schmid。VectorNet：从向量化表示中编码高精地图和智能体动态。发表于 CVPR，2020年。2[13] Dan Hendrycks 和 Kevin Gimpel。高斯误差线性单元（GELU）。arXiv，2016年。12[14] Yizhou Huang、Yihua Cheng 和 Kezhi Wang。Trajectory Mamba：基于选择性状态空间模型的高效注意力-Mamba预测模型。发表于 CVPR，2025年。6, 7[15] Peter J Huber。位置参数的稳健估计。《数理统计年刊》，1964年。12[16] Xiaosong Jia、Penghao Wu、Li Chen、Yu Liu、Hongyang Li 和 Junchi Yan。HDGT：通过场景编码实现多智能体轨迹预测的异构驾驶图Transformer。IEEE TPAMI，2023年。2[17] Zhiqian Lan、Yuxuan Jiang、Yao Mu、Chen Chen、Shengbo Eben Li、Hang Zhao 和 Keqiang Li。SEPT：迈向高效的运动预测场景表征学习。发表于 ICLR，2023年。2、4、11[18] Ming Liang、Bin Yang、Rui Hu、Yun Chen、Renjie Liao、Song Feng 和 Raquel Urtasun。学习用于运动预测的车道图表示。发表于 ECCV，2020年。2[19] Yicheng Liu、Jinghuai Zhang、Liangji Fang、Qinhong Jiang 和 Bolei Zhou。使用堆叠Transformer进行多模态运动预测。CVPR，2021年。2[20] Ilya Loshchilov 和 Frank Hutter。解耦权重衰减正则化。发表于 ICLR，2019年。13[21] Jean Mercat、Thomas Gilles、Nicole El Zoghby、Guillaume Sandou、Dominique Beauvois 和 Guillermo Pita Gil。用于多模态联合车辆运动预测的多头注意力。发表于 ICRA，2020年。2[22] Nigamaa Nayakanti、Rami Al-Rfou、Aurick Zhou、Kratarth Goel、Khaled S Refaat 和 Benjamin Sapp。Wayformer：通过简单高效的注意力网络进行运动预测。发表于 ICRA，2023年。2[23] Jiquan Ngiam、Benjamin Caine、Vijay Vasudevan、Zhengdong Zhang、Hao-Tien Lewis Chiang、Jeffrey Ling、Rebecca Roelofs、Alex Bewley、Chenxi Liu、Ashish Venugopal 等。SceneTransformer：用于预测多个智能体轨迹的统一架构。ICLR，2022年。2[24] Ziqi Pang、Deva Ramanan、Mengtian Li 和 Yu-Xiong Wang。面向自动驾驶的流式运动预测。发表于 IEEE智能机器人与系统国际会议，2023年。2、3、6、15[25] Alexander Prutsch、Horst Bischof 和 Horst Possegger。高效运动预测：一种轻量且准确的轨迹预测模型，具有快速的训练和推理速度。发表于 IEEE智能机器人与系统国际会议，2024年。2、3、4、6、11[26] Charles R Qi、Hao Su、Kaichun Mo 和 Leonidas J Guibas。PointNet：用于三维分类和分割的点集深度学习。发表于 CVPR，2017年。2、4[27] Luke Rowe、Martin Ethier、Eli-Henry Dykhne 和 Krzysztof Czarnecki。FJMP：基于学习得到的有向无环交互图进行因式分解联合多智能体运动预测。发表于 CVPR，2023年。7[28] Shaoshuai Shi、Li Jiang、Dengxin Dai 和 Bernt Schiele。结合全局意图定位与局部运动细化的运动Transformer。发表于 NeurIPS，2022年。1、2、3、4、6[29] Shaoshuai Shi、Li Jiang、Dengxin Dai 和 Bernt Schiele。MTR++：采用对称场景建模和引导式意图查询的多智能体运动预测。IEEE TPAMI，2024年。2、6、8[30] Nan Song、Bozhou Zhang、Xiatian Zhu 和 Li Zhang。连续驾驶中的运动预测。发表于 NeurIPS，2024年。2、3、4、5、6、7、8、11、13、14、15、16、17、18[31] Xiaolong Tang、Meina Kan、Shiguang Shan、Zhilong Ji、Jinfeng Bai 和 Xilin Chen。HPNet：结合历史预测注意力的动态轨迹预测。发表于 CVPR，2024年。2、3

[32] Balakrishnan Varadarajan、Ahmed Hefny、Avikalp Srivastava、Khaled S Refaat、Nigamaa Nayakanti、Andre Cornman、Kan Chen、Bertrand Douillard、Chi Pang Lam、Dragomir Anguelov 等。MultiPath++：用于行为预测的高效信息融合与轨迹聚合。发表于 ICRA，2022年。2[33] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。Attention Is All You Need。发表于 NeurIPS，2017年。4[34] Mingkun Wang、Xinge Zhu、Changqian Yu、Wei Li、Yuexin Ma、Ruochun Jin、Xiaoguang Ren、Dongchun Ren、Mingxu Wang 和 Wenjing Yang。GANet：用于运动预测的目标区域网络。发表于 ICRA，2023年。2、6[35] Shihao Wang、Yingfei Liu、Tiancai Wang、Ying Li 和 Xiangyu Zhang。探索面向高效多视角三维目标检测的以物体为中心的时序建模。发表于 CVPR，2023年。5[36] Xishun Wang、Tong Su、Fang Da 和 Xiaodong Yang。ProphNet：使用锚点引导提议的高效以智能体为中心的运动预测。发表于 CVPR，2023年。6[37] Benjamin Wilson、William Qi、Tanmay Agarwal、John Lambert、Jagjeet Singh、Siddhesh Khandelwal、Bowen Pan、Ratnesh Kumar、Andrew Hartnett、Jhony Kaesemodel Pontes、Deva Ramanan、Peter Carr 和 James Hays。Argoverse 2：面向自动驾驶感知与预测的新一代数据集。发表于 NeurIPS数据集与基准，2021年。1、2、5、6、11、13、14[38] Wenyuan Zeng、Ming Liang、Renjie Liao 和 Raquel Urtasun。LaneRCNN：用于以图为中心的运动预测的分布式表示。发表于 IEEE智能机器人与系统国际会议，2021年。2[39] Bozhou Zhang、Nan Song 和 Li Zhang。DeMo：将运动预测解耦为方向意图和动态状态。发表于 NeurIPS，2024年。2、3、4、6、7、8、11、13、14、15[40] Lu Zhang、Peiliang Li、Sikang Liu 和 Shaojie Shen。SIMPL：用于自动驾驶的简单高效多智能体运动预测基线。IEEE机器人与自动化快报，2024年。6[41] Zhejun Zhang、Alexander Liniger、Christos Sakaridis、Fisher Yu 和 Luc Van Gool。通过采用相对位姿编码的异构折线变换器实现实时运动预测。发表于 NeurIPS，2023年。2、6[42] Yang Zhou、Hao Shao、Letian Wang、Steven L. Waslander、Hongsheng Li 和 Yu Liu。SmartRefine：一种场景自适应的高效运动预测细化框架。发表于 CVPR，2024年。1、2、3、4、6、7、8[43] Zikang Zhou、Jianping Wang、Yung-Hui Li 和 Yu-Kai Huang。以查询为中心的轨迹预测。发表于 CVPR，2023年。1、2、3、6、7、13[44] Zikang Zhou、Zihao Wen、Jianping Wang、Yung-Hui Li 和 Yu-Kai Huang。QCNeXt：用于联合多智能体轨迹预测的新一代框架。arXiv，2023年。7

使用端点感知建模的实时流式轨迹预测

补充材料

在本补充材料中，我们首先在第A节比较流式处理与传统的基于快照的方法。为确保可复现性，接下来我们将介绍模型的全部实现细节（第B节），以及如何将其应用于多智能体场景（第C节）。我们还将详细介绍评估（第D节），展示其他结果，例如针对恢复早期预测误差的鲁棒性研究，并提供我们方法的可视化结果（第E节）。最后，我们将概述代码框架（第F节），该框架也包含在补充材料中。

A. 流式处理与基于快照的处理

图5 比较了标准的基于快照的轨迹预测范式与我们及相关工作采用的基于流式的处理方式 [30, 39]。两种范式使用相同的场景划分来区分训练数据和验证数据，从而确保公平比较。为了在缺少更长上下文数据的情况下实现流式处理，我们减少了输入模型的历史输入。这样便可在同一组训练数据中执行多次预测。

基于流式的处理与真实世界的部署场景十分接近，在这些场景中，轨迹预测模型以连续设置运行。它能在连续的预测步骤之间实现信息流动，不同于每次预测都独立进行的基于快照的范式。这样一来，模型便可以利用时间信息，从而实现更稳健且一致的预测。

对于 Argoverse 2 [37]，我们将模型输入历史长度设为$h = 3$秒，这对应于 $T_h = 30$ 个输入样本；并在 $t \in \{3, 4, 5\}$秒处执行三次预测，每次间隔 1 秒。按照基准设置，我们预测 6 秒的未来，这对应于 $T_f = 60$ 个输出时间步。为了在流式处理中充分利用可用的训练数据，我们还会额外预测未来 $T_a = 20$个时间步。因此，模型总输出时长为 8 秒。对于所有评估以及端点中心特征的选择，只有未来 T_f个时间步相关。在第一次和第二次模型轮次中，额外的未来时间步 T_a 使我们能够在更长的未来时间范围上计算回归损失，从而有效地教会模型预测长期未来。在针对 Argoverse 1 [3] 的消融实验中，我们使用 1 秒的历史输入（$T_h = 10$），并在 $t \in \{1, 1.5, 2\}$秒处执行三轮。

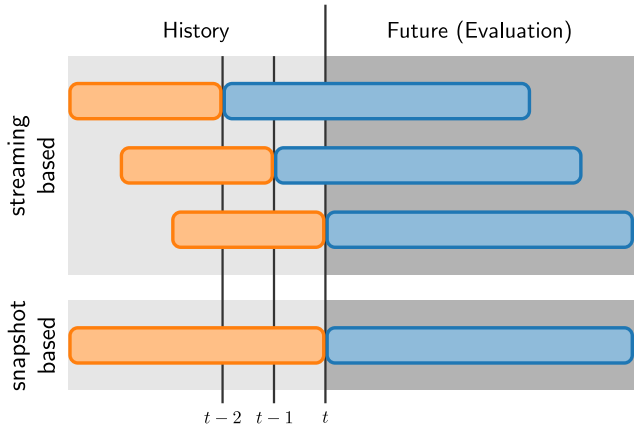

图5. 基于快照和基于流式处理的轨迹预测范式对比。两种方法均使用相同的基准输入数据，确保在不引入额外数据的情况下进行公平比较。在流式范式中，过去的观测通过滑动窗口进行处理，这与实际部署条件十分接近。为模拟真实世界运行中的挑战，流式处理中建立了信息传递机制。与独立处理每一帧的基于快照的处理相比，这种方式能够实现更一致、时间连贯性更好的预测。在本例中，流式模型第三轮处理得到的预测可以直接与基于快照的方法得到的预测进行评估。

B. 实现细节

我们将模型的特征维度设为 $D = 128$。类别类型嵌入用于区分四种智能体类型（车辆、行人、骑行者和其他）以及三种车道类型（标准车道、自行车道和公交车道）。我们使用标准的 150 米半径来收集场景元素 [4,25, 30, 39]。为了控制场景标记的数量，我们不将车道拆分为更小的车道段，例如例如 [11, 17]所建议的做法。我们使用 xy位置、 xy速度以及相应的有效标志来编码智能体数据，得到 $D_a = 5$。为了编码车道数据，我们使用每个车道点的xy位置，并使用有效标志屏蔽超出感兴趣区域的车道段部分，得到$D_l = 3$。

B.1. 位置嵌入

为了对所有元素的全局位姿进行建模——包括智能体、车道和以目标为中心的坐标系——我们用各自的位置（x, y）和旋转 yaw 来表示它们。对于智能体，我们使用其初始位置和朝向；对于车道，我们使用中心线的中心点和朝向；对于以目标为中心的坐标系，我们使用轨迹端点

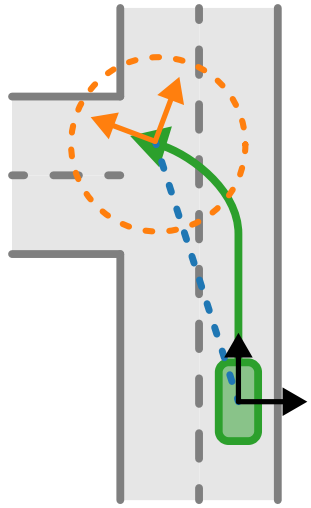

图6. 以目标为中心的坐标系（橙色）的建模，使用前一时间帧的预测（绿色）。我们还展示了目标上下文半径（橙色）以及用于建模其与以智能体为中心的原点（黑色）之间关系的位置嵌入（蓝色）。

并根据轨迹末端的曲率计算朝向。

B.2. 以目标为中心的坐标系

图6 展示了我们以目标为中心的坐标系的设置。我们使用前一时间步的轨迹预测终点 F^{t-1}，创建一个以目标区域为中心并围绕其定向的局部参考坐标系。随后，使用以目标为中心的坐标系对目标上下文半径内的所有场景特征进行编码。为捕捉全局空间关系，我们还加入从以智能体为中心的原点到以目标为中心的坐标系的位置嵌入（蓝色虚线）。

B.3. 多头注意力块

我们的多头注意力块采用以下配置：

- 注意力头数量：8
- 丢弃率：0.2
- 前馈扩展因子：$4 \cdot D$
- 激活函数：GELU [13]
- 注意力操作前归一化

B.4. 编码器

我们在智能体编码器中使用四个编码器块 f_A 并在我们的场景上下文编码器中使用四个编码块 f_S。对于我们的目标上下文编码器 f_T 我们使用两个编码器块。表6 对我们新颖的目标上下文编码器的深度进行了消融研究。由于词元数量 N_c' 在我们的

数据集	# 块	mADE_6	mFDE_6	b-mFDE_6
AV2 验证集	1	**0.66**	1.26	1.86
	2	**0.66**	**1.25**	**1.85**
	3	**0.66**	1.26	**1.85**

表 6. 我们的目标编码器中不同数量注意力块的消融研究 f_T。

以目标为中心的上下文小于以智能体为中心的上下文中的词元数量N_c，因此可以使用更浅的编码器。为了计算给定位姿（x，y，θ）的位置嵌入，我们首先将其变换为（x，y，$\sin\theta$，$\cos\theta$）。然后，我们应用一个两层多层感知机（多层感知机）：

- 第 1 层：$4 \times D$
- GELU [13]激活函数
- 第 2 层：$D \times D$

B.5. 解码器

我们的双上下文解码器包含三个阶段（$d = 3$），总计包含六个交叉注意力块。如果智能体没有先前的预测，则会跳过对以目标为中心的上下文执行的交叉注意力步骤。用于处理解码后的查询 Q' 并输出未来轨迹 F 的多层感知机包含两层：

- 第 1 层：$D \times 2 \cdot D$
- ReLU 激活函数
- 第 2 层：$2 \cdot D \times 2 \cdot (T_f + T_a)$

类似地，用于输出概率分数的多层感知机 P：

- 第 1 层：$D \times 2 \cdot D$
- ReLU 激活函数
- 第 2 层：$2 \cdot D \times 1$

如上所述（第 A节），我们的解码器会预测 T_a额外的步数，除去作为评估中实际轨迹的 T_f 步。

B.6. 优化

我们采用常见的赢家通吃策略来训练模型 [4],，仅使用拟合效果最佳的预测轨迹（即平均位移误差最小的轨迹）进行优化。我们使用平滑 L1 损失（Huber 损失 [15]）作为回归损失 L_{reg}，以确保假设与真实值相符。此外，我们使用标准交叉熵损失 L_{cls}，为拟合效果最佳的轨迹赋予最高置信度分数。为了进一步提升学习效果，我们采用辅助损失 L_{aux} [4]：为场景中的每个非焦点智能体预测一条轨迹，并应用平滑 L1 损失。为了预测辅助未来轨迹，我们针对场景上下文 S中除焦点智能体以外、对应于智能体的每个 token 使用一个线性层 $D \times 2 \cdot T_f$。最终损失为 $L = L_{\text{reg}} + L_{\text{cls}} + L_{\text{aux}}$。

我们训练模型 80 轮次，前 13 轮次作为预热阶段，在此期间，我们将学习率从 1e-5 线性提高到 1e-2。随后，

我们使用单一余弦退火调度将学习率降至 1e-5。训练时在单张 NVIDIA Quadro RTX 8000 上使用批量大小为 32。我们采用 AdamW [20]作为优化器，使用基于范数的梯度裁剪，最大值设为 5，并将权重衰减设为 1e-2。我们只使用 Argoverse 2 训练集进行训练，不进行预训练或数据增强。

C. 多智能体扩展

C.1. 实现细节

在 AV2 多智能体设置中，目标是预测所有评分智能体的轨迹。我们的全局一致性模块使用两个 Transformer 块，在智能体的所有模式上进行自注意力；随后使用两个 Transformer 块，在每种模式下跨智能体（世界）进行自注意力。第一阶段，我们对每个智能体的解码结果$Q' \in \mathbb{R}^{N_{a,\ s} \times K \times D}$在所有模式上进行自注意力，其中$N_{a,\ s}$是评分智能体的数量。接下来，我们将查询置换为 $\mathbb{R}^{K \times N_{a,\ s} \times D}$，并在每种模式下跨所有智能体进行自注意力 $N_{a,\ s}$。为了更好地建模不同的智能体行为，我们在 Q'中加入类别类型嵌入，以区分焦点智能体、驾驶中的评分智能体，以及可能处于停放状态的评分智能体（根据边缘预测判断）。我们使用一个两层多层感知机生成世界预测 $F_w \in \mathbb{R}^{K \times N_{a,\ s} \times T_f \times 2}$：

- 第 1 层：$D \times 2 \cdot D$
- ReLU 激活函数
- 第 2 层：$2 \cdot D \times 2 \cdot T_f$

类似地，用于输出相关置信度分数的多层感知机 $P_w \in \mathbb{R}^K$：

- 第 1 层：$D \times 2 \cdot D$
- ReLU 激活函数
- 第 2 层：$2 \cdot D \times 1$

我们采用与单智能体场景相同的流式处理设置。

C.2. 优化

为了将我们的方法应用于 Argov- erse 2 (AV2) 中的多智能体场景，[37], 我们使用单智能体模型（边际预测模型）初始化模型权重。随后，我们在 AV2 多智能体训练集上端到端联合训练边际预测模型和全局一致性模块，共训练 35 轮，不设预热阶段。为减少内存消耗，我们冻结智能体编码器和车道编码器。训练期间，我们采用余弦退火调度，将学习率从 1e-2 降至 1e-5。

同样，我们采用赢家通吃原则，仅优化最佳世界（平均最小位移误差最低）中的轨迹。与单智能体训练类似，我们对最佳世界中的每个智能体预测使用回归损失 L_{reg}，并采用交叉熵分类损失 L_{cls}，将最高置信度分数赋予最匹配的世界。为保持较强的单智能体性能并引导多智能体学习，我们还在多智能体场景的训练过程中加入单智能体损失 L_{marginal}（见上文）。最终损失表示为 $L = L_{\text{reg}} + L_{\text{cls}} + L_{\text{marginal}}$。

D. 评估详情

D.1. 指标

我们使用标准 AV2 基准指标评估所提出的方法。每项指标均基于排名前 k 的轨迹假设进行评估。最小平均位移误差（minADE$_k$）是所有时间步中真实值与最拟合假设之间欧几里得距离的平均值。最小最终位移误差（minFDE$_k$）仅考虑最后一个时间步的距离；而 Brier 最小最终位移误差（brier-minFDE$_k$）则在 minFDE$_k$ 上添加惩罚项 $(1-\pi)^2$，其中 π 是最拟合轨迹的概率分数。漏检率（MR$_k$）用于评估是否存在预测终点与真实值终点之间距离在 2 米以内的情况。

在多智能体场景中，我们基于排名前 k 的世界评估所有指标。每个世界都为每个参与评分的 智能体包含一条未来轨迹。平均最小平均位移误差（avgMinADE$_k$）通过对一个世界中所有行动体的 minADE$_k$ 取平均值计算得出。类似地，平均最小最终位移误差（avgMinFDE$_k$）对所有行动体的 minFDE$_k$ 取平均值；平均布里尔最小末端位移误差（avgBrierMinFDE$_k$）则对一个世界中所有行动体的 brier-minFDE$_k$ 取平均值。行动体漏检率 actorMR$_k$ 表示所有评分智能体中，终点与真实值终点相距 2 米以内的智能体所占的比例。

D.2. 延迟测量

我们使用官方代码实现来测量 QCNet[1] [43], RealMotion[2] [30], 和 DeMo[3] [39]的延迟。为确保公平比较，并消除不同数据加载和预处理实现带来的影响，我们仅测量模型前向传播的推理延迟。

E. 其他结果

E.1. AV2 验证集上的流式方法

表 7 比较了在 Argoverse 2 验证集上基于流式处理的轨迹预测方法。我们的方法再次取得了最佳的 Brier-minFDE$_6$, 该指标同时考虑了位移误差和轨迹评分。通过比较$_6$ 不同模型的 Brier-minFDE$_6$ 与 minFDE$_6$ 值，我们可以看出，我们的方法在估计

[1] https://github.com/ZikangZhou/QCNet[2]
https://github.com/fudan-zvg/RealMotion[3]
https://github.com/fudan-zvg/DeMo

方法	minADE_1	minFDE_1	MR_6	minADE_6	minFDE_6	brier-minFDE_6
RealMotion [30]	1.65	4.10	0.16	0.67	1.30	1.94
DeMo [39]	**1.48**	**3.73**	**0.13**	**0.61**	**1.19**	1.86
SEAM（我们的方法）	1.60	3.96	0.15	0.66	1.25	**1.85**

表 7. 在 Argoverse 2 验证集上针对单智能体轨迹预测的基于流式处理的方法比较。对于所有模型，我们报告的结果均未采用模型集成。

终点噪声	mADE_6	mFDE_6	b-mFDE_6
$\mathcal{N}(0,\ 5)$	0.67	1.28	1.88
$\mathcal{U}(-5,\ 5)$	0.67	1.27	1.87
$\mathcal{N}(0,\ 3)$	0.67	1.26	1.87
$\mathcal{U}(-3,\ 3)$	0.67	1.26	1.86
$\mathcal{N}(0,\ 1)$	0.67	**1.25**	**1.85**
$\mathcal{U}(-1,\ 1)$	0.67	**1.25**	**1.85**
None	**0.66**	**1.25**	**1.85**

表8. 通过扰动 2 [37] s 时的预测端点，评估我们提出的端点感知建模在 AV $t \in \{3, 4\}$验证集上的鲁棒性。这些端点用于定义提取以目标为中心的特征时所用的锚点，并通过加性均匀噪声（$\mathcal{U}$）或高斯噪声（$\mathcal{N}$）进行修改。所示结果对应于 $t = 5$s 时的预测误差。

预测轨迹的可能性方面表现出色，这对于解读自运动规划等下游任务中的结果至关重要。

E.2. 对误差传播的鲁棒性

表 8评估了在先前预测存在噪声时，我们提出的用于信息流的端点感知建模的鲁棒性。为此，我们对用于获取以目标为中心的特征的端点锚点添加均匀噪声或高斯噪声。结果表明，在预测噪声较小时，性能仅略有下降。尽管如此，即使过去的预测受到误差影响，我们的双上下文方法仍能保持稳健的性能。以目标为中心的特征提供了额外信息，可提升整体预测准确性，同时不限制预测，使模型能够在当前时间步的以智能体为中心的特征指示不同的未来运动时进行恢复。随着噪声幅度增加，性能进一步下降；然而，即使存在较大的噪声偏移，模型仍具有很强的韧性。

此外，我们还在图 7中分析了误差的变化。图中比较了在场景开始 $_6$ 秒时生成的首次预测的 minFDE $t = 3$ 与在 $t = 5$ 秒时生成的最终预测，两者均在 6 秒的未来时间范围内进行评估。结果表明，我们的双上下文解码方法能够从早期预测误差中恢复：即使初始预测不准确（横轴上的值较高），最终预测仍可能非常准确（值较低

在纵轴上）。也存在一些初始预测准确、但最终预测性能下降的情况，例如发生意外机动时，例如突然停车，

在第一个预测窗口中并不明显，但在第二个窗口中变得相关。总体而言，模型的预测质量随时间推移而提升，这一点从 $x = y$ 对角线下方更高的点密度可以看出。

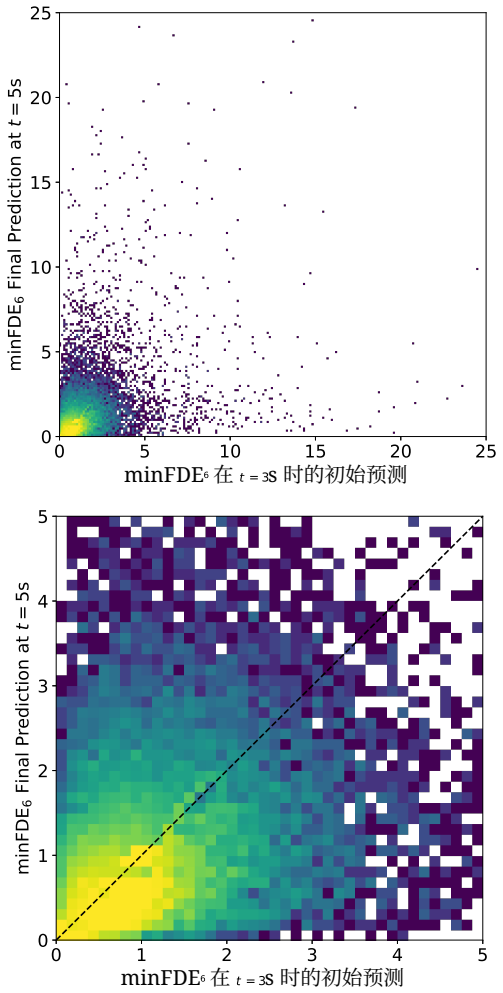

图 7. 预测误差的密度图。两幅图均比较了在 $t = 3$s 时生成的首次预测（横轴）与在 $t = 5$ s 时生成的最终预测（纵轴）。上图显示，即使初始误差较大，模型仍能恢复并修正其预测。下图放大展示了原点附近的区域。位于 $x = y$ 对角线（虚线）下方的点密度较高，表明预测通常会随着时间推移而改善。

方法	延迟（$B = 1$）		延迟（$B = 32$）		延迟（$B = 64$）	
	离线	在线	离线	在线	离线	在线
RealMotion [30]	51 毫秒	**13 毫秒**	255 毫秒	85 毫秒	512 毫秒	172 毫秒
DeMo [39]	73 毫秒	18 毫秒	223 毫秒	71 毫秒	433 毫秒	143 毫秒
SEAM（我们的方法）	**50 毫秒**	**13 毫秒**	**95 毫秒**	**39 毫秒**	**185 毫秒**	**65 毫秒**

表 9。使用一块 NVIDIA A10 GPU 预测 B Argoverse 2 单智能体场景的延迟分析。我们比较了基于流式处理的方法，并报告离线和在线推理延迟。在线延迟与实际应用相关。

模型	训练数据	全局一致性模块	行动体MR$_6$	avgMinADE$_6$	avgMinFDE$_6$	avgBrierMinFDE$_6$
RealMotion [30]	单智能体数据	✗	0.727	1.413	4.060	4.716
SEAM（本文方法）	单智能体数据	✗	0.693	1.324	3.643	4.256
SEAM（本文方法）	在多智能体数据上微调	✗	0.228	0.720	1.619	2.126
SEAM（本文方法）	在多智能体数据上微调	✓	**0.155**	**0.600**	**1.180**	**1.814**

表10. 将单智能体方法适配到 AV2 多智能体验证集的消融研究。前两行表明，在没有任何模型适配或微调的情况下，我们的方法和相关工作 [30] 在多智能体场景中的表现都不佳。在多智能体数据集上进行微调可显著提升性能，而加入用于场景级评分的全局一致性模块则能取得最佳结果。

方法	波动
RealMotion [30]	0.347
SEAM（我们的方法）	**0.341**

表11。我们将模型的轨迹波动与 RealMotion 进行比较 [30]。由 [24] 定义的波动指标可用于衡量轨迹在多个时间步上的一致性。

E.3. NVIDIA A10 GPU 上的延迟

我们在表中展示了使用一块 NVIDIA A10 GPU 得到的更多延迟结果，并对比了基于流式处理的方法 9。在这款较新的 GPU架构上，我们的方法同样取得了最佳延迟表现，在所有批量大小下均实现了最低的在线和离线延迟。

E.4. 多智能体消融研究

我们在表10中对将单智能体方法扩展到多智能体场景进行了消融研究。前两行表明，直接将单智能体模型应用于多智能体基准会导致性能不佳。这主要是因为多智能体数据集包含单智能体数据无法充分表征的运动模式。此外，这些方法忽略了智能体预测之间的全局一致性。若没有专门的全局一致性模块，我们会将每个智能体最可能的轨迹组合成一个世界、次可能的轨迹组合成另一个世界，以此类推，从而生成多个合理的未来世界。在多智能体数据上微调模型有助于捕捉更广泛的行为（第 3 行），而加入显式的全局一致性模块则能进一步提升性能，取得最佳结果（第 4 行）。

E.5. 轨迹波动

表11 比较了我们的方法与 RealMotion 的轨迹波动情况 [30]。波动指标按照 [24] 的定义，用于衡量多个预测帧之间轨迹的一致性。与 RealMotion 相比，我们的方法取得了更低的波动分数，表明其预测更加一致。

E.6. 结果可视化

我们在图中展示了 Argoverse 2 验证集场景的更多定性结果8 和图 9。

E.7. 失败案例

图10展示了我们的方法无法正确预测未来轨迹的失败案例。通常，失败是由于智能体在预测时刻做出了无法预料的运动所致，也常常与地图数据不足有关，即缺少对私人车道的建模。

F. 代码实现

为提升清晰度并便于复现，我们还提供了代码实现。随附的自述文件说明了如何配置工作环境，以及如何针对单智能体任务执行训练、验证和可视化。代码库还包含多智能体场景的所有实现（文件名带有 ma 前缀或后缀）。论文录用后，我们将发布源代码和预训练模型。我们的代码基于 RealMotion[2] 的实现。

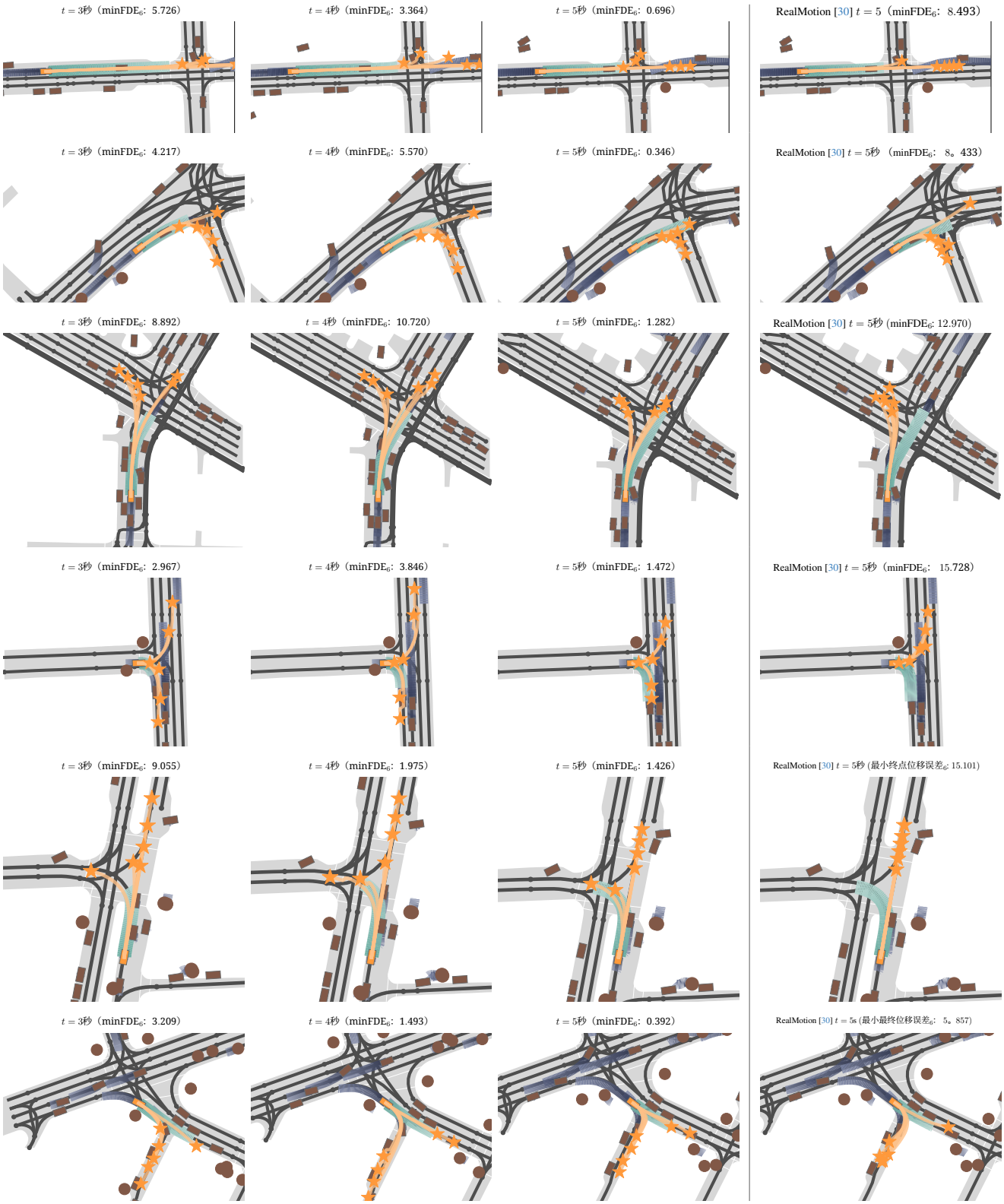

图8. 我们的方法在 Argoverse 2 验证集场景中的定性结果。图中展示了预测，即流式处理方法在 $t \in \{3,4,5\}$秒时的结果。可视化还展示了未来真实值、智能体历史轨迹以及邻近智能体。右侧列展示了在流式设置下使用 RealMotion [30]时，于 $t=5$秒得到的最终预测。

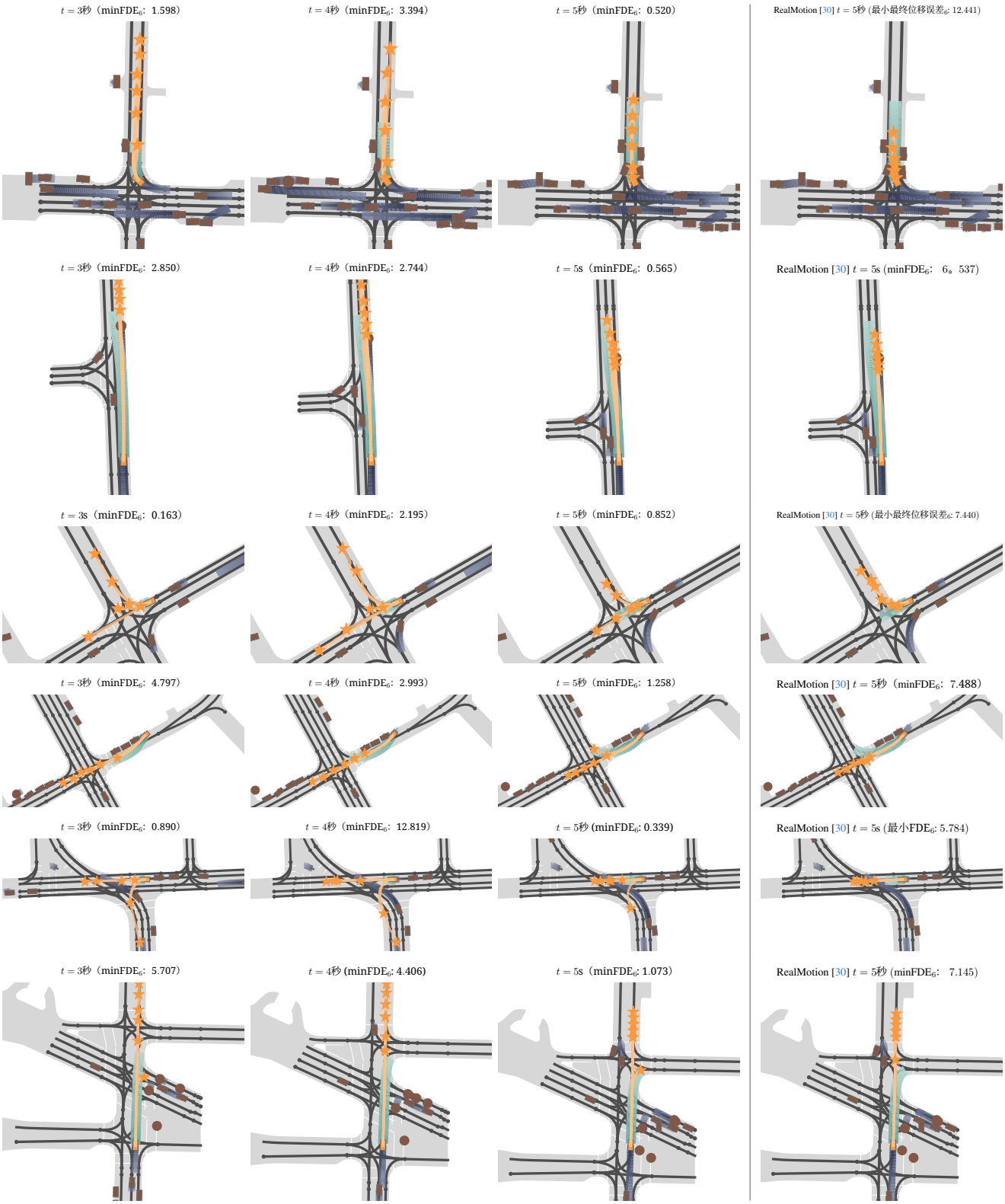

图9. 我们的方法在 Argoverse 2 验证集场景上的定性结果。我们展示了**预测**，这些结果由我们的流式处理方法在 $t \in \{3, 4, 5\}$秒时生成。可视化还显示了**未来真实值**、**智能体历史轨迹**和**邻近智能体**。右列展示了在流式设置下使用 RealMotion [30] 时于 $t = 5$秒生成的最终预测。

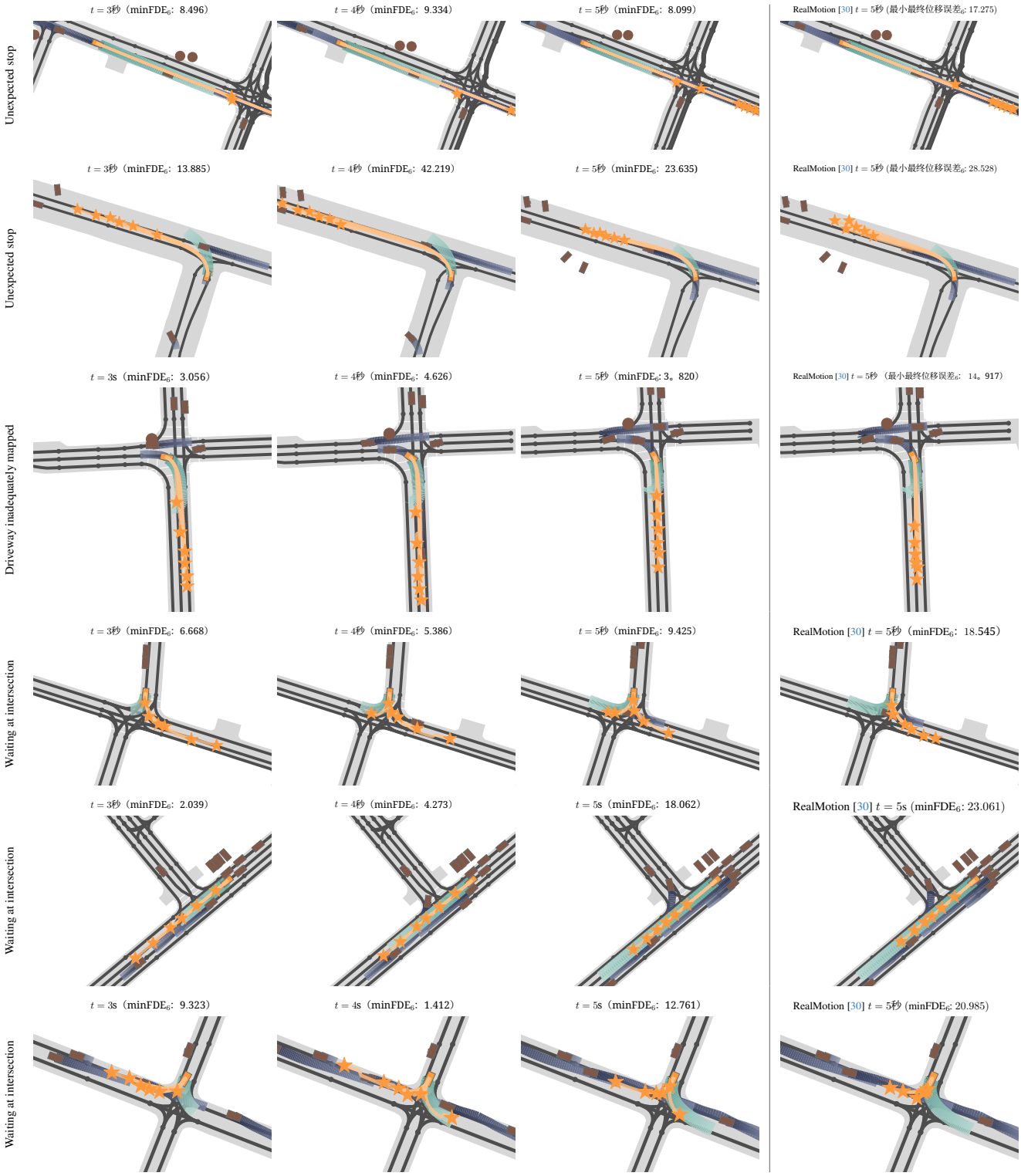

图 10. 我们的方法在 Argoverse 2 验证集场景中的失败案例。我们展示了 **基于流式处理的方法在** $t \in \{3, 4, 5\}$ 秒时的**预测**。可视化还展示了**未来真实值**、**智能体历史轨迹**和邻近智能体。右侧列展示了在流式设置下使用 RealMotion [30] 时，于 $t = 5$ 秒生成的最终预测。

在第一个场景中，一辆车辆在距离较远的交叉路口前停下，原因目前尚不明确。在第二个场景中，一辆车辆因当时未被检测到的原因停下。在第三个场景中，车辆开始转入一条未纳入车道数据建模的私人车道。第四、第五和第六个场景展示了车辆在交叉路口等待的情形，其未来路径并不明确。