Check for updates

Original Research Article

Institution of MECHANICAL ENGINEERS
Journal of AUTOMOBILE ENGINEERING

Proc IMechE Part D:
J Automobile Engineering
1–21
© IMechE 2025

Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/09544070251397454
journals.sagepub.com/home/pid
Sage

Goal oriented trajectory prediction conditioned on reachable road context

Chenhui Yin[1], Marco Cecotti[2] and Haobin Jiang[3]

Abstract

Trajectory prediction of surrounding traffic agents is crucial for autonomous vehicles to perform collision-free and efficient planning at urban intersections. Despite interactions with neighbour objects, road layout information plays an essential role in improving prediction accuracy and enhancing the interpretability of prediction models. However, exploring reachable areas and effectively leveraging these contextual clues in predictions remain challenging. In this work, a goal-oriented trajectory prediction framework is proposed to integrate valuable road layout information. The framework leverages sparse and non-uniform map elements to represent moving intentions. For effective exploration of relevant map elements, a constrained breadth-first search is proposed, enabling simultaneous and efficient exploration across lateral and longitudinal directions by incorporating behavioural constraints. The attention mechanism and a dynamic mask are combined to focus on the most relevant map element features and predict corresponding goal points, facilitating the final trajectory prediction. This progressive narrowing of the inference space enhances both the accuracy and interpretability of the prediction model. Experimental results on the Intersection Drone Dataset and Roundabout Drone Dataset demonstrate that the proposed model achieves a 73.1% accuracy in predicting the most likely map elements, with an average displacement error below 0.75 m and a final displacement error around 1.95 m with a prediction horizon of 4 s.

Keywords

trajectory prediction, breadth-first-search, goal-oriented, attention mechanism, drone dataset

Received: 24 January 2025; accepted: 28 October 2025

Introduction

Accurate long-term trajectory prediction of surrounding traffic agents is crucial for autonomous vehicles to navigate complex driving scenarios safely, particularly at urban intersections where traffic dynamics are highly interactive and uncertain.[1] Extensive research has focused on modelling interactions between target agents (TAs) and surrounding agents (SAs) using deep learning (DL) techniques, such as Convolutional Neural Networks (CNNs),[2] Graph Neural Networks (GNNs)[3] and the Attention Mechanism (AM),[4–6] which effectively capture spatial-temporal dependencies. Concurrently, it is equally important to integrate geometric and semantic information of road layouts (RL), which can guide predictions by following lane geometry, connectivity, and other constraints influencing the TA's manoeuvres.[7,8] However, urban intersections present unique challenges that have not been adequately addressed by existing approaches, hindering the integration of RL information into prediction backbones.

Urban intersections are usually characterised by sparse vectorised map representations, where map elements spatially overlap despite being topologically distant. This spatial-topological discrepancy poses distinct difficulties in efficient and lightweight reachable area exploration and destination-oriented context extraction.

Related work

The influence of RL maps can be evaluated by adding map-aware features as additional inputs, for example, the TA's deviation from the lane centreline or lane

[1]School of Automotive and Traffic Engineering, Jiangsu University, Zhenjiang, China
[2]Advanced Vehicle Engineering Centre, Cranfield University, Bedford, UK
[3]Automotive Engineering Research Institute, Jiangsu University, Zhenjiang, China

Corresponding author:
Marco Cecotti, Advanced Vehicle Engineering Centre, Cranfield University, Bedford MK43 0AL, UK.
Email: m.cecotti@cranfield.ac.uk

markings.[9] Though easy to implement, this implicit feature mining approach can lead to a large and heterogeneous input vector, making it difficult for monolithic NNs to efficiently extract and abstract road-related features.

One alternative is to discretise the road scene around the TA into rasterisation images or multiple-channel grid maps in Bird-Eye-View (BEV) and apply CNNs for spatial feature extraction.[10–13] These image-like representations are intuitive and preserve local traffic scene contexts.[14] Nevertheless, they struggle with representing long-range topologies, are often storage-intensive and can be less efficient in real-time applications.[3,15,16] Besides, it is hard to realise a topology-guided feature exploration.

A more structured approach is to organise the RL map as a graph, where each node corresponds to a lane segment and edges are built based on agents' traversability across lane segments. A lane segment is usually represented by a sequence of ordered polyline points, either from centrelines or boundaries.[15,17] This vectorised representation is sparser than image-based methods while retaining the RL topology and the map graph allows message passing along the RL topology using GNNs.[18] For example, Liang et al.[3] aggregate vehicle and lane node embeddings and apply a Graph Convolutional Network (GCN) to propagate this contextual information over the map graph. Similar techniques have been effectively applied in literature, enabling global topology information to be captured.[7,19,20]

However, to reach a topologically far node, there is a risk of over-smoothing while performing multi-step message propagation along the map graph, where node features tend to become indistinguishable after multiple aggregation steps and the quality of feature representation degrades.[3] Another issue is that relating the TA to the most relevant lane nodes can be non-trivial when multiple spatially overlapping lane segments are present, potentially leading to incorrect agent-lane assignments. A wrong match might cause incorrect feature concatenation because overlapped lane nodes are close in the Euclidean space, but far from each other in the RL topology. Moreover, many existing works combine RL features with those of the TA in a black-box manner and introduce data from irrelevant RL elements, providing limited interpretability regarding how RL information informs the TA's predicted manoeuvres.

To address these limitations, some studies have proposed two-stage approaches.[8] The first stage aims to identify the lanes and lane points that are relevant to the prediction, that is, that are likely to be reached within the prediction horizon, starting from the given position of the TA. These generated intermediate outputs can be interpreted as the intention of TA and are then used to guide the final prediction in the following stage.

Lane instances within a customised distance centred at the TA are usually relevant, where lane candidates are selected by a range search (RngS). With these prior selected lanes, Pan et al.[21] calculate their importance through MLP based on the historical and current deviations from the TA to the lane centrelines. In contrast, Luo et al.[22] and Liu et al.[23] calculate their probability of reachable lanes by dot-product based on the TA's historical encoding and lane encodings. With candidate lanes and their scores, AMs are usually used to fuse the encodings of the TA and of the candidate lanes, which further contributes to trajectory prediction. A significant advantage of AM over GCN is that the features of individual lanes are preserved fully, without smoothing due to message propagation. However, most existing methods derive comprehensive lane element encodings solely through a weighted sum of relevant encodings without providing explanations for the assigned weights. This can lead to the selection of features from irrelevant lane instances, as evidenced by the work of Wang et al.[24] While the model may still operate effectively under these conditions, it does not ensure learning from ideal data, like the ground truth, thereby potentially compromising interpretability.

To further narrow down the attention scope, lane pieces instead of complete lanes are selected for lane encoding abstraction. To get a concise map of interest, Gómez et al.[25] select relevant lanes based on the RL topology and then prune them according to the TA's current position and an estimation of its future travel distances. Taking advantage of a graph representation of the RL, Gao et al.[26] perform depth-first-search (DFS) to identify relevant lane nodes starting from the currently occupied node. In the DFS, the maximum depth is limited according to the average speed of the TA. Kim et al.[17] apply an RngS with a range of 10 m to identify potential candidate lane segments and then use DFS to extend these lane segments by searching their predecessors and successors until a predefined distance is reached. In this combination, DFS is used to extend exploration in the longitudinal direction of the lanes while RngS is used to compensate for searching in the traversal direction. Though plausible lanes are identified, redundant information is included in the reverse moving direction and the lane representation is too rigid to describe manoeuvres like lane changing.

An alternative for selecting lanes or lane pieces is to estimate the goal point of the TA within the prediction horizon. For this purpose, Gu et al.[27] select sample points on the best candidate lane and then use AM to select the candidate goal point. Lu et al.[19] pre-sample anchor points in the drivable area evenly and predict their probability directly, of which the top K points are selected to do final predictions. Gilles et al.[28] select the most likely lane segments before estimating goal points within the chosen lane segments.

By estimating goal points according to RL maps, moving intentions have been embodied and interpretability has been improved. However, these methods rely either on pre-sampled anchor points as candidates[19,27] or the discretisation of selected lane pieces to estimate

goal points.[28] These operations are storage-intensive and limit the expression ability of NNs because of restricting the outputs of NNs to a discretisation space.

Motivations and contributions

Existing goal-oriented trajectory prediction methods rely heavily on continuous lane structures to illustrate primary motion intentions,[17,22,23] such as complete lane instances, consecutive lane segments, or immediate successors. These elements originate near the current position of the TA and extend to distant locations. However, these approaches lack flexibility in identifying destinations involving lane-changing manoeuvres, as continuous lane elements inherently restrict exploration to fixed pathways.

To overcome this limitation, sparse, fragmented lane segments—sometimes even distant ones—are used to represent coarse destinations, thereby avoiding reliance on predefined continuous lane structures.

When exploring the Area of Interest (AOI), existing search strategies often struggle to balance longitudinal lane-following and lateral lane-changing exploration. For example, commonly used methods like RngS indiscriminately include lane segments within a predefined circular radius.[22,29] This approach introduces lateral redundancy, as it depends on homogeneous Euclidean distance metrics, whereas effective longitudinal exploration typically requires larger radii. Conversely, DFS leverages the map's graph structure to facilitate efficient longitudinal exploration within a specified distance threshold.[26] However, supplementary lateral exploration mechanisms are still necessary in lane-changing scenarios. Methods such as RngS or K-Nearest Neighbours (KNN) are often combined with DFS to identify neighbouring lanes,[17,25] followed by longitudinal extension of the AOI along these selected lateral segments. Unfortunately, this hybrid search framework imposes additional computational and storage burdens.

To address these identified limitations, this research focuses on map data mining for accurate, efficient and interpretable trajectory prediction at urban intersections. A systematic framework is developed, grounded in domain-specific constraints. A unified Constrained Breadth-First Search (cBFS) approach is proposed to balance longitudinal search efficiency and lateral search relevance through a simultaneous process. To prevent excessive lateral exploration, a lateral search depth constraint is proposed based on realistic vehicle manoeuvring capabilities and statistical analysis of lane-change manoeuvres, which transforms BFS into a domain-adapted tool.

Building upon this refined AOI exploration, a goal-oriented trajectory prediction framework is introduced. This framework infers the moving intentions of TAs by scoring lane segments that the TA will reach in the AOI and subsequently estimating potential goal points, progressively narrowing down the search space.

Rather than propagating information across all reachable elements within AOI, AM is employed to establish a direct supervisory link between the traffic context and TA coarse intentions. This is achieved by estimating the most relevant map element and undergoing explicit training on ground-truth destination elements. However, cBFS may fail to identify an adequate number of map elements in specific edge cases, such as at map boundaries or when TAs operate off-road. To address these cases, a dynamic map mask is introduced that assigns a zero score to vacant positions in the AOI. This effectively excludes the influence of irrelevant map elements, enhances adaptability to both on-road and off-road agents, and stabilises the training process.

Subsequently, a goal estimation is performed based on context mining results from the map elements within the AOI. Unlike existing works that either pre-sample goal candidates across drivable areas or rely on the fusion of scenario features without explicit grounding in map elements,[30,31] the potential goal point is generated by focusing on the most influential map elements. This approach reduces the impact of less relevant map elements, an aspect often neglected in related studies.

The novel contributions of this paper are as follows:

(1) A novel goal-oriented trajectory prediction framework is proposed. This framework leverages fragmented map elements to represent coarse moving destinations and employs a flexible map element mask to exclude the influence of less relevant elements and enhance the adaptability to TAs not following RLs, thereby ensuring stable performance.
(2) A unified AOI exploration method, denoted as cBFS, is proposed based on BFS. cBFS incorporates behavioural lane-change constraints to facilitate effective and efficient simultaneous longitudinal and lateral exploration with lightweight storage. This mechanism eliminates the need for separate lateral/longitudinal search algorithms and ensures balanced and efficient exploration aligned with real-world driving constraints.

This work is significant to the intelligent transportation and autonomous driving communities because:

(1) It provides an approach to efficiently mine road scene and interaction information, enhancing the prediction accuracy.
(2) The interpretability of the prediction is improved by systematically refining the inference space from coarse map elements to goal points. The probabilities associated with the top map elements and their corresponding goal points provide insight into the model's understanding of moving intentions and subsequent destinations.
(3) The proposed RL map exploration method can be adjusted with customised accumulated

distance and lane-changing limitations, and thus can be generalised in various scenarios.

In the rest of this paper, Section 'Methodology' presents the methodologies to get data ready for model inputs. In Section 'Prediction model,' the goal-oriented trajectory prediction model is proposed. Experiments and results are reported and discussed in Section 'Experiment evaluation', followed by the conclusions drawn in Section 'Conclusions'.

Methodology

The problem of trajectory prediction in this paper is formulated as determining the future trajectories of a TA conditioned on its positional data and corresponding useful scenario information (including neighbours and RL maps) at each discrete time step t over a fixed-size historical horizon. Given observations $\boldsymbol{X}$ in the form of past statuses and scenario info of the TA, the prediction model seeks to provide its future trajectory. In this research, the TA includes cars, buses and bicycles in urban scenarios, consisting of cross intersections and a roundabout.

Inputs and outputs

Formally, a set of observable features $\boldsymbol{X}$ and a set of target outputs $\hat{\boldsymbol{Y}}$ to be predicted are considered. It is assumed that all features can be accessed simultaneously at each time step and that historical measurements with a sequence length of $h + 1$ are available, where h is a fixed time horizon. Let $\mathcal{T}_{\text{hist}} = \{0, \ldots, h\}$ and, for $k \in \mathcal{T}_{\text{hist}}$, the value of the TA's status k time steps earlier is denoted by $\boldsymbol{X}^{(t-k)}$. Similarly, define $\mathcal{T}_{\text{pred}} = \{0, \ldots, f\}$ and, for $k \in \mathcal{T}_{\text{pred}}$, denote the value of output time steps in the future by $\hat{\boldsymbol{Y}}^{(t+k)}$. It is proposed to use a DL-based model, in which a regression function F is trained such that the predicted outputs $\hat{\boldsymbol{Y}} = F(\boldsymbol{X})$ match the actual values $\boldsymbol{Y}$ as closely as possible.

The inputs to the model are a set of track histories and surrounding traffic information of the TA:

$$\boldsymbol{X} = \left[\boldsymbol{X}^{(t-h)}, \quad \ldots, \quad \boldsymbol{X}^{(t-1)}, \quad \boldsymbol{X}^{(t)} \right], \tag{1}$$

And at a time instant t, for the TA, the following features are defined in $\boldsymbol{X}^{(t)}$:

- TA's information $\boldsymbol{TA}$, including absolute position in the x and y direction with a local Cartesian coordinate and velocities in the x and y directions, denoted by p_x, p_y, v_x and v_y respectively,
- SAs' information $\boldsymbol{SA}$, defined in equation (5) of Section 'Neighbour information',
- Map information $\boldsymbol{M}_{\text{cnl}}$, defined in equation (6) of Section 'Map information',

$$\boldsymbol{X}^{(t)} = [\boldsymbol{TA}^{(t)}, \boldsymbol{SA}^{(t)}, \boldsymbol{M}_{\text{cnl}}^{(t)}]. \tag{2}$$

Though velocity information has been implicitly included in the position values as a constant time interval is applied, the velocity values are used as additional inputs to the model input.

The output of the model is the predicted positions over the prediction horizon:

$$\hat{\boldsymbol{Y}} = \left[\hat{\boldsymbol{Y}}^{(t+1)}, \quad \ldots, \quad \hat{\boldsymbol{Y}}^{(t+f)} \right], \tag{3}$$

and at each time frame t,

$$\hat{\boldsymbol{Y}}^{(t)} = [p_x{}^{(t)}, p_y{}^{(t)}] \tag{4}$$

represents the x and y coordinates at the time t of the TA being predicted.

Neighbour information

It is common to arrange SAs in a grid which is centred at the TA and aligned to the TA's moving direction. However, this rectangular grid shape does not conform to the actual road layout and is more expensive in storage in case it needs to be extended to cover SAs that do not move in the same direction as the TA, like in intersections. Considering this limitation, SAs in this research are identified by a range search and any surrounding agent within a range of 60 m is considered an SA.

This includes both static and kinetic objects and makes it more adaptable to complex traffic environments.

SAs' information at the time t is stored as below:

$$\boldsymbol{SA}^{(t)} = \begin{bmatrix} p_{x_1}^{(t)}, p_{y_1}^{(t)}, v_{x_1}^{(t)}, v_{y_1}^{(t)}, \ldots \\ p_{x_k}^{(t)}, p_{y_k}^{(t)}, v_{x_k}^{(t)}, v_{y_k}^{(t)}, \ldots \\ p_{x_n}^{(t)}, p_{y_n}^{(t)}, v_{x_n}^{(t)}, v_{y_n}^{(t)} \end{bmatrix}, \tag{5}$$

where $k \in [1, n]$ and n denotes the maximum number of SAs. This is a fixed-size vector and the features of individual SAs are grouped together. The maximum number of SAs is set to be 15.[29] If an SA is absent in the vector, zeros are filled instead.

Map information

This work focuses on urban intersections. Road layout and geometric information are provided in the Lanelet2 format.[32] In this representation, an atomic map element, like a piece of lane segment, is called a lanelet and is defined by sequential points of its two boundaries. These boundary points can be low-density, which contributes to a sparse and storage-efficient

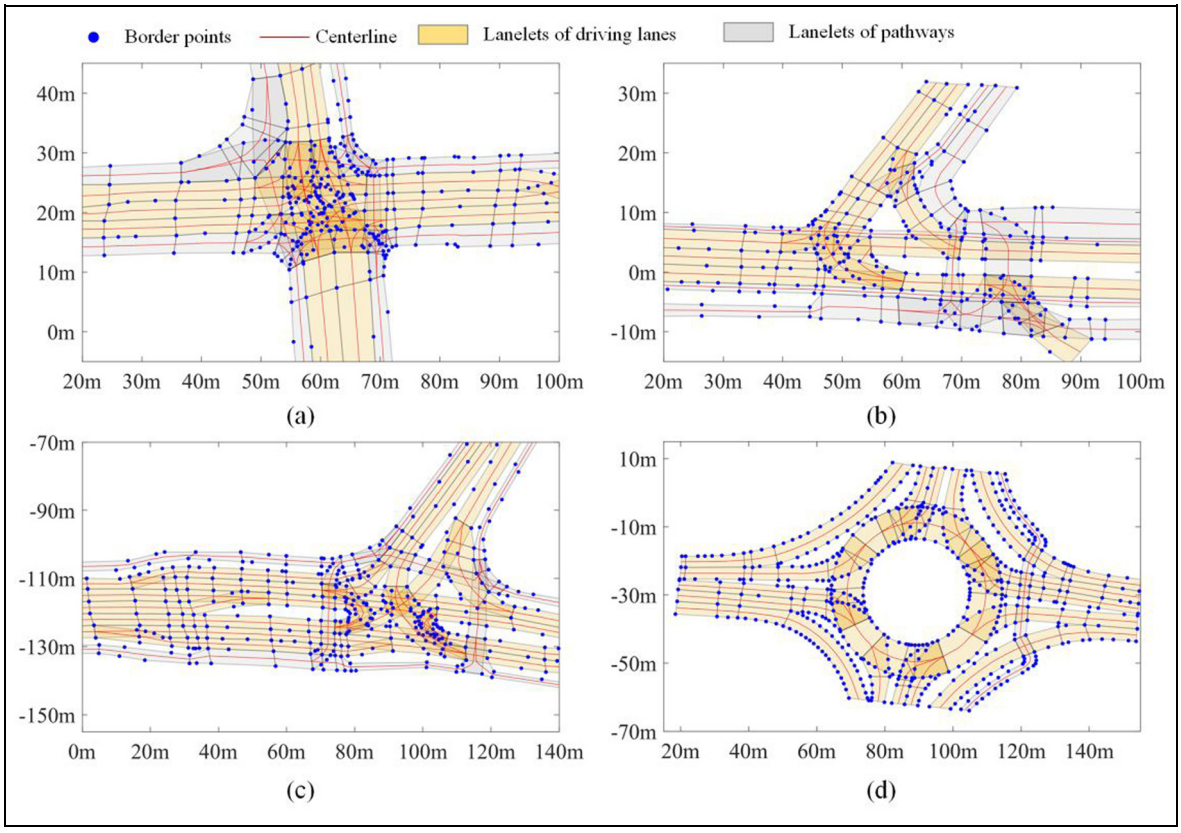

Figure 1. The target intersections are formatted in Lanelet2. Boundary points of lanelets are marked in blue and their centrelines are depicted in red. Lanes are represented by connected lanelets. Driving lanes and pathways are shown in light yellow and grey, respectively.

description. Totally, three interactions from the inD dataset (Intersections Drone Dataset)[33] and one roundabout from the rounD dataset (Roundabouts Drone Dataset)[34] are included.

Map data representation. For a more compact representation and easier extraction of map element features, the centrelines of lanelets are used instead of their boundaries in the description of road topologies and geometries. The centrelines are calculated by searching for maximal disks along the lanelets, which produces robust and smooth outputs.[35,36] An illustration of Lanelet2 maps of the target scenarios used in this research is shown in Figure 1.

The centreline of a lanelet consists of several ordered centre points. To facilitate the application of CNN-based map feature extraction in Section 'Map element feature encoder', each centreline is down-sampled from a dense point representation and is composed of 18 centre points, the sequence of which is aligned with the driving direction.

With these definitions, the map $\boldsymbol{M}_{\text{cnl}}$ is an assembly of all lanelet centrelines in the target scenario, each consisting of multiple centre points represented by their coordinates:

$$\boldsymbol{M}_{\text{cnl}} = [cnl_1, cnl_2, \ldots, cnl_i, \ldots, cnl_m\,], \tag{6}$$

where $i \in [0, m]$ and cnl_i represents the centreline of the ith map element. m denotes the number of the map elements and its value is 600 in this work, determined by the total number of map elements exported from the Lanelet2 maps of all four target scenarios.

Constrained breadth-first search. In the identification of potential map elements that the TA might travel through, cBFS is proposed to limit the searching distance and the search in the traversal direction.

cBFS extends to the children, left and right neighbours according to the RL, starting from the map element occupied by the TA. During cBFS, the search depth is constrained to travel distances. To realise this, the distance from the current position to the last centre point of the reachable lanelet is accumulated, and once the cumulated distance d_{cum} exceeds the predefined limit, cBFS is terminated.

The distance limit $d_{\text{cum_max}}$ is defined as follows:

$$d_{\text{cum_max}} = v_{\text{limit}} \cdot t_{\text{pred}} \cdot N\,, \tag{7}$$

where v_{limit} is the speed limit in the scenario (13.9 m/s), t_{pred} is the duration of the prediction horizon and N is a redundancy factor considering potential over-speeding. The value of N is determined in Section 'Results of

Table 1. The pipeline of cBFS.

Algorithm **1** Map element exploration based on cBFS

Data:
- A graph for map elements $\boldsymbol{G}_{\text{map}}$: a Struct: map elements are stored in rows and indexed by their row locations (idx_{elem}) in the Struct. Each row contains the row locations of the children, left and right neighbours and the corresponding map element length l_{elem}.
- Maximum accumulated distance $d_{\text{cum_max}}$ and the maximum number of allowed lane-changing manoeuvres $N_{\text{LC_max}}$.

Input:
- The index of the map element $idx_{\text{elem_strt}}$, where the cBFS starts

Initialization:
- Mark all map elements as unvisited: $\boldsymbol{V}[idx_{\text{elem}}] = 0$.
- Create an empty set $\boldsymbol{M}_{\text{reachable}} = []$ to store indexes of all reachable map elements.
- Initialise a queue $\boldsymbol{Q}_{\text{cand}}$ to store candidate map elements with initial values: $[idx_{\text{elem_strt}}, d_{\text{cum}} = 0, N_{\text{LC}} = 0]$.
- Mark the start map element as visited: $\boldsymbol{V}[idx_{\text{elem_strt}}] = 1$.

Perform cBFS:
while $\boldsymbol{Q}_{\text{cand}}$ is not empty:
  Dequeue the first element in $\boldsymbol{Q}_{\text{cand}}$:
    $[idx_{\text{elem_curr}}, d_{\text{cum}}, N_{\text{LC}}]$.
  Add $idx_{\text{elem_curr}}$ into $\boldsymbol{M}_{\text{reachable}}$.
  Retrieve children as candidate map elements in
    $\boldsymbol{G}_{\text{map}}$, along with their lengths l_{elem}.
  if $N_{\text{LC}} < N_{\text{LC_max}}$:
    Retrieve left and right neighbours as candidate map
    elements in $\boldsymbol{G}_{\text{map}}$, along with their lengths l_{elem}.
    Update the N_{LC}: $N_{\text{LC}} = N_{\text{LC}} + 1$.
  end

  for candidate map elements:
    Compute the accumulated distance as:
      $d_{\text{cum}} = d_{\text{cum}} + l_{\text{elem}}$.
    if $\boldsymbol{V}[idx_{\text{elem}}] = 0$ and $d_{\text{cum}} < d_{\text{cum_max}}$
      Enqueue $[idx_{\text{elem_cand}}, d_{\text{cum}}, N_{\text{LC}}]$.
      Mark the candidate as visited:
        $\boldsymbol{V}[idx_{\text{elem_cand}}] = 0$.
    end
  end
end
Return: $\boldsymbol{M}_{\text{reachable}}$

cBFS' based on speed distribution analysis in the target scenarios.

This longitudinal searching ensures a deep enough exploration to cover potential map elements while saving storage space compared to the DFS.[26] In addition, compared to methods that extract the whole driving lane, this approach includes lane-changing manoeuvres.

While BFS inherently supports topology-aware traversal, unconstrained lateral expansion leads to excessive exploration and degraded AOI relevance. Thereby, the maximum exploration depth along the traversal direction (left and right neighbours) is constrained by a maximum number of lane changes (LC): $N_{\text{LC_max}}$. This is grounded in the empirical observation that TAs exhibit limited lane-change capability within a 4 s prediction horizon. The value of $N_{\text{LC_max}}$ is determined in Section 'Results of cBFS', considering the feasible number of lane changes in the prediction horizon.

Through these two search limitations, the BFS is constrained in both the longitudinal distance and traversal depth. The pipeline of cBFS is shown in Table 1.

Attention mechanism

AM operates based on the target and source matrices. Each vector t_i in the target matrix $\boldsymbol{T}_{\text{attn}} = [t_1, t_2, \ldots t_{tn}]$ will attend to all vectors in the source matrix $\boldsymbol{S}_{\text{attn}} = [s_1, s_2, \ldots s_{sn}]$ and create a comprehensive encoding vector. In the application, these two matrices will be converted into the query, key and value matrices.

The query matrix is generated from $\boldsymbol{T}_{\text{attn}}$:

$$\boldsymbol{Q}_{\text{attn}} = \boldsymbol{T}_{\text{attn}}\boldsymbol{W}^{Q} = \left[q^1, \ldots, q^{tn-1}, q^{tn}\right], \tag{8}$$

where tn is the number of query vectors, $\boldsymbol{W}^{Q}$ is the learnable parameter performing linear projection to get the query vectors and is shared among all vectors in $\boldsymbol{T}_{\text{attn}}$.

The key and value matrices are generated similarly by:

$$\boldsymbol{K}_{\text{attn}} = \boldsymbol{S}_{\text{attn}}\boldsymbol{W}^{K} = \left[k^1, \ldots, k^{sn-1}, k^{sn}\right], \tag{9}$$

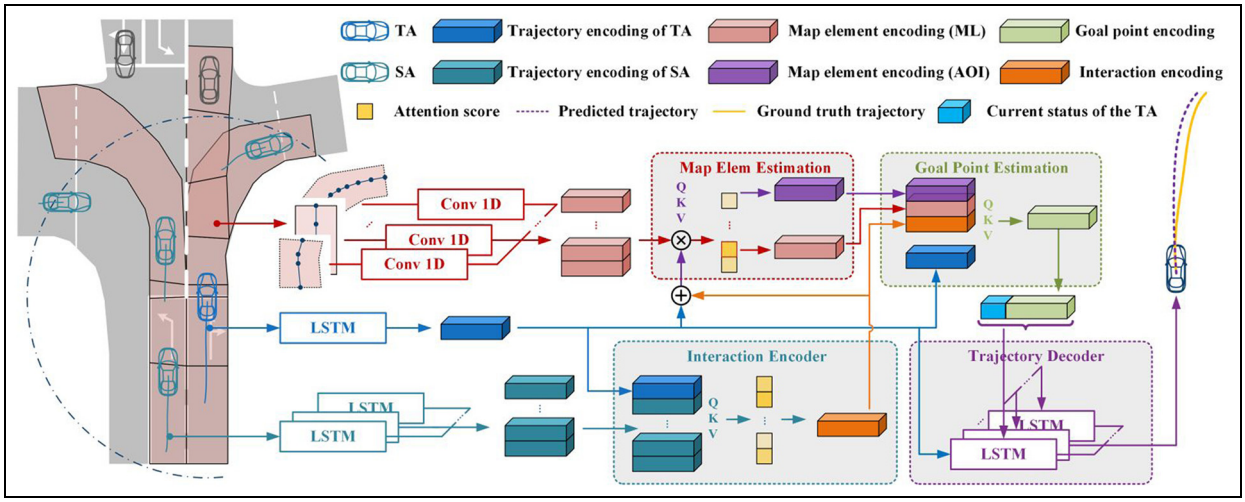

Figure 2. An LSTM-based encoder-decoder backbone is used to process the historical trajectories and generate the predictions. The SAs are identified by a range search and their historical trajectories, as well as that of the TA, are embedded by the LSTM. With these trajectory encodings, an AM-based Interaction encoder is used to mine interaction clues between the TA and SAs. The reachable map elements are abstracted by 1D convolution. The most likely map element that the TA will reach is identified by AM in the map element estimation module and the corresponding map feature encoding is extracted, as well as the encoding of AOI. Another AM-based goal point estimation module is then applied to fuse the scene context encodings, including the map and interaction encodings, and estimate the final goal. At last, LSTM is used to generate the final predictions with a concatenation of goal point encoding and the current state.

$$\boldsymbol{V}_{\text{attn}} = \boldsymbol{S}_{\text{attn}}\boldsymbol{W}^{V} = \left[v^{1}, \ldots, v^{sn-1}, v^{sn}\right], \tag{10}$$

where sn is the number of vectors, $\boldsymbol{W}^{K}$ and $\boldsymbol{W}^{V}$ are learnable parameters performing linear projection to get the key and value vectors and are shared among vectors in $\boldsymbol{S}_{\text{attn}}$.

The scaled dot product is used to calculate the correlation score $\boldsymbol{A}_{\text{attn}}$ between the query and key vectors:[37]

$$\boldsymbol{A}_{\text{attn}} = M_{\text{attn}}(\boldsymbol{Q}_{\text{attn}}, \boldsymbol{K}_{\text{attn}}) = \text{softmax}\left(\frac{\boldsymbol{Q}_{\text{attn}}\boldsymbol{K}_{\text{attn}}^{T}}{\sqrt{d_k}}\right) \tag{11}$$

where $M_{\text{attn}}(\cdot)$ is the Matrix-matrix operation, d_k is the dimension of the query vector.

The attention context is a weighted sum based on the query, key and value vectors:

$$\boldsymbol{H}_{\text{attn}} = \text{Attn}(\boldsymbol{Q}_{\text{attn}}, \boldsymbol{K}_{\text{attn}}, \boldsymbol{V}_{\text{attn}}) = \boldsymbol{A}_{\text{attn}} \cdot \boldsymbol{V}_{\text{attn}}. \tag{12}$$

AM can be classified into self-attention and cross-attention. If the target matrix $\boldsymbol{T}_{\text{attn}}$ and source matrix $\boldsymbol{S}_{\text{attn}}$ are different, then it is cross-attention. If the target matrix $\boldsymbol{T}_{\text{attn}}$ and source matrix $\boldsymbol{S}_{\text{attn}}$ are the same, it is self-attention. [38,39]

Prediction model

Contrary to many existing frameworks for intent or behaviour prediction, which can be modelled as classification problems, the aim of this research is to predict future positions for the TA across a prediction horizon, which is intrinsically a regression problem. The proposed model is shown in Figure 2.

Trajectory encoder

The historical track of each agent (including the TA and SAs) is encoded by using an LSTM encoder. This module is widely applied in extracting features of sequential inputs.[37,38,40] At any time instant t, a sequence of h time steps of the trajectory history are passed through the encoder. The LSTM states for each agent are updated frame by frame over the h past frames. The LSTM weights are shared across the sequences of all agents. The final LSTM state for each agent i can be expected to encode the motion status of that agent. The sequential inputs are:

$$\boldsymbol{D}_i = \left[\boldsymbol{D}_i^{(t-h)}, \quad \ldots, \quad \boldsymbol{D}_i^{(t-1)}, \quad \boldsymbol{D}_i^{(t)}\right]. \tag{13}$$

At any time instant t,

$$\boldsymbol{D}_i^{(t)} = \left[p_{x_i}^{(t)}, p_{y_i}^{(t)}, v_{x_i}^{(t)}, v_{y_i}^{(t)}\right], \tag{14}$$

$$\boldsymbol{h}_{\text{hist}}^{i}, \boldsymbol{c}_{\text{hist}}^{i} = \text{LSTM}(\boldsymbol{D}_i, (\boldsymbol{h}_{\text{hidden}}, \boldsymbol{h}_{\text{cell}})), \tag{15}$$

where $\boldsymbol{h}_{\text{hist}}^{i}$ and $\boldsymbol{c}_{\text{hist}}^{i}$ are the final hidden state and cell state of the LSTM, respectively. They have the same shape with a hidden size of D_{LSTM}. $\boldsymbol{h}_{\text{hidden}}$ and $\boldsymbol{h}_{\text{cell}}$ are the initialisation value of the LSTM hidden and cell states, which are vectors of zeros. In the equation, i is

the index of the agents where 0 for the TA and positive integers for SAs.

In the neighbour interaction encoder, self-attention will be used to mine interaction information among the TA and SAs, as shown in Section 'Neighbour interaction encoder'. To keep the consistency, the same LSTM encoder is shared among the TA and SAs.

Neighbour interaction encoder

AM is used to mine the interaction between the TA and SAs. Self-attention is used; thus, the target and source matrices are designed to be the same and to include historical encodings of both the TA and SAs. In this case, the TA learns not only from the SAs' encoding but also from itself.[41] Besides, the interaction among SAs could also be learned.

The agent interaction attention context is represented by:

$$\boldsymbol{H}_{\text{nbr}} = \text{Attn}(\boldsymbol{H}_{\text{hist}}\boldsymbol{W}^{Q}_{\text{hist}}, \boldsymbol{H}_{\text{hist}}\boldsymbol{W}^{K}_{\text{hist}}, \boldsymbol{H}_{\text{hist}}\boldsymbol{W}^{V}_{\text{hist}}), \quad (16)$$

where $\boldsymbol{H}_{\text{nbr}}$ is the generated agent-agent attention context with a length of $n+1$, $\boldsymbol{H}_{\text{hist}}$ is the aggregation of $\boldsymbol{h}^{i}_{\text{hist}}$ for both TA and SAs with a length of $n+1$, and $\boldsymbol{W}^{Q}_{\text{hist}}$, $\boldsymbol{W}^{K}_{\text{hist}}$, and $\boldsymbol{W}^{V}_{\text{hist}}$ are learnable parameters performing linear projection.

TA's encoding is appended after the aggregation of SAs' encodings; thus the last element of $\boldsymbol{H}_{\text{nbr}}$ corresponds to the TA's encoding fused with neighbour interaction context, denoted as $\boldsymbol{h}_{\text{nbrCtx}}$.

In the source matrix, the neighbours are stored in a position-invariant way and their orders will not influence the attention results. To avoid training instability, encodings of zeros in the source matrix are used when SAs are absent.

Map element feature encoder

The map element feature encoder aims to extract geometry information, such as curvature, length and direction, by processing the original map element centre points.

A group of 1D convolution (1D Conv) layers and max-pooling layers are used to produce an implicit representation of the map element features. 1D Conv is usually used in encoding sequential data, such as past trajectories of agents[18,19] and ordered lane points,[17] and is also reported as efficient and lightweight. The map element feature encoder is illustrated in Figure 3. A group of 1D Conv and max pooling layers are shared among all map elements. By this map element feature encoder, the presentation of each map element is converted from a sequence of centrepoints to an array of high-dimensional features.

Table 2. Shape of outputs after each layer in the map element feature encoder.

Layers	Shape of output
Input	$(m, 2, 18)$
1D Conv_1	$(m, D_{\text{conv_1}}, 18)$
Max Pooling	$(m, D_{\text{conv_1}}, 8)$
1D Conv_2	$(m, D_{\text{conv_2}}, 6)$
Max Pooling	$(m, D_{\text{conv_2}}, 3)$
1D Conv_3	$(m, D_{\text{conv_2}}, 1)$

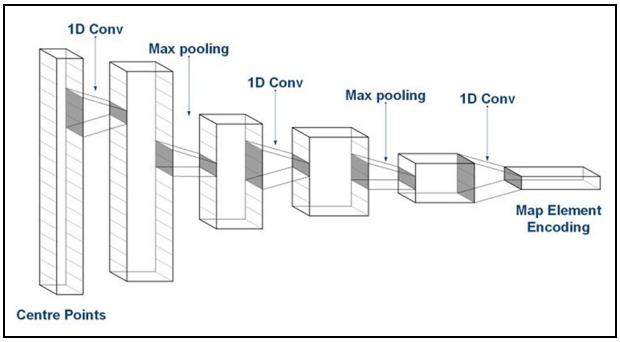

Figure 3. The map element feature encoder. Three layers of 1D Conv and two layers of max pooling are used to convert the centre points' coordinates to a high-dimensional map element encoding.

The input is the map element feature matrix $\boldsymbol{M}_{\text{cnl}}$ with shape $(m, D_{\text{pos}}, n_{\text{cpt}})$, where m denotes the number of map elements, D_{pos} denotes the dimension of centre point features (x and y coordinates), and is the number of centre points in a single map element, which is 18 in this research.

The map element features are computed as:

$$\boldsymbol{M}_{\text{conv_1}} = \text{maxPl}(\text{Conv_1}(\boldsymbol{M}_{\text{cnl}})), \quad (17)$$

$$\boldsymbol{M}_{\text{conv_2}} = \text{maxPl}(\text{Conv_2}(\boldsymbol{M}_{\text{conv_1}})), \quad (18)$$

$$\boldsymbol{M}_{\text{sce}} = \text{Conv_3}(\boldsymbol{M}_{\text{conv_2}}), \quad (19)$$

where Conv_$i \in \{1, 2, 3\}$ denotes 1D convolution layers and maxPl$(\cdot)$ represents the max pooling operation. Each 1D Conv layer uses a kernel size of 3, followed by a max-pooling layer with a kernel size of 2. All 1D Conv layers are followed by Leaky ReLU activation, omitted from equations for brevity. In addition, no padding is used in 1D Conv layers. Shapes of map element features after each layer are summarised in Table 2 below, which is also aligned with the illustration in Figure 3.

In Table 2, $D_{\text{conv_1}}$, $D_{\text{conv_2}}$ and $D_{\text{conv_3}}$ stand for channel dimensions of three 1D Conv layers, respectively.

Most likely map element estimator

The most likely (ML) map element to cover the final point of the trajectory is selected based on the co-relationship calculated by the TA's historical encoding and map element feature encodings through cross-attention. This is inspired by LAformer,[23] while the scene representation and module details are different. Rather than just generating a comprehensive map-aware encoding, the attention scores between the TA and map elements are first computed.

The map element selection is jointly determined by historical status and interactions with SAs. Practically, the TA's historical encoding is first concatenated with the neighbour interaction encoding:

$$\boldsymbol{h}_{\mathrm{mQ}} = \mathrm{cat}\left(\boldsymbol{h}^{0}_{\mathrm{hist}}, \boldsymbol{h}_{\mathrm{nbrCtx}}\right), \tag{20}$$

This joint encoding serves as the query vector, with the key-value pairs derived from map element feature encodings. The agent-map attention context is formulated as:

$$\boldsymbol{A}_{\mathrm{attnM}} = \mathrm{Attn}\left(\boldsymbol{h}_{\mathrm{mQ}}\boldsymbol{W}^{Q}_{\mathrm{map}}, \boldsymbol{M}_{\mathrm{sce}}\boldsymbol{W}^{K}_{\mathrm{map}}\right), \tag{21}$$

where $\boldsymbol{W}^{Q}_{\mathrm{map}}$ and $\boldsymbol{W}^{K}_{\mathrm{map}}$ are learnable parameters performing linear projection.

The map element with the highest attention score is selected as the most likely map element and its map element encoding is therefore extracted, denoted as $\boldsymbol{M}^{\mathrm{ML}}_{\mathrm{sce}}$, for goal point estimation (seen in Section 'Goal point estimator').

Additionally, a comprehensive context is derived from the AOI through:

$$\boldsymbol{h}_{\mathrm{AOI}} = \boldsymbol{A}_{\mathrm{attnM}} \cdot \boldsymbol{M}_{\mathrm{sce}}\boldsymbol{W}^{V}_{\mathrm{map}}, \tag{22}$$

where $\boldsymbol{W}^{V}_{\mathrm{map}}$ represents the learnable parameters for linear projection to generate value vectors and $\boldsymbol{h}_{\mathrm{AOI}}$ denotes a fused encoding integrating map element features within the AOI, with a hidden size of D_{mapAttn}.

Map element features from all scenarios are aggregated into $\boldsymbol{M}_{\mathrm{sce}}$. Without distinguishing map elements across scenarios or considering reachability, the TA may inappropriately reference all elements in the container to compute weights, potentially misleading learning and causing ambiguity for DL-based models. Additionally, cBFS can identify insufficient map elements in the AOI due to edge cases, for example, at the margin of the map. To handle these cases, a dynamic map mask is introduced.

In detail, features of map elements within the AOI, explored via cBFS, are extracted from $\boldsymbol{M}_{\mathrm{sce}}$ and are stored in $\boldsymbol{M}_{\mathrm{sce_AOI}}$. Let m_{AOI} denote the maximum number of map elements allowed in the AOI. If fewer than m_{AOI} elements are extracted via cBFS, dummy map elements of zero-value features are appended to ensure matrix completeness of $\boldsymbol{M}_{\mathrm{sce_AOI}}$. A local mask is then created to flag these dummy map elements in AOI:

$$\boldsymbol{Mask}_{\mathrm{AOI}}[i] \in \{0,1\}^{1\times m_{\mathrm{AOI}}}. \tag{23}$$

where $\boldsymbol{Mask}_{\mathrm{AOI}}$ assigns 1 to all dummy elements. Based on experimental validation, m_{AOI} is set to be 40 to balance data coverage and computational efficiency.

During the attention process in equation (21), logits for map elements in AOI are computed via the dot product. Prior to Softmax normalisation, logits corresponding to masked map elements are set to a negligible constant, for example, −9e15, ensuring their probabilities effectively become zero after Softmax, thus restricting selection to valid AOI elements.

For edge cases where no valid AOI elements are detected, the dummy element's logit is set to a large value (e.g., 100). This forces the dummy element to receive a score of 1 with all others scoring 0, resolving ambiguity. This is particularly useful for cases like bicycles that may not adhere to lane structures and no map elements are matched, which has been neglected by published works based on Argoverse since only vehicles' trajectories are predicted. Practically, bicycle map element encodings are zero-padded, as they often operate outside predefined road structures. Our approach can identify these cases and predict a trajectory accordingly.

Goal point estimator

The potential goal point of the prediction is closely related to scene context, which includes both map information ($\boldsymbol{h}_{\mathrm{AOI}}$, $\boldsymbol{M}^{\mathrm{ML}}_{\mathrm{sce}}$) and interactions between neighbouring agents ($\boldsymbol{h}_{\mathrm{nbrCtx}}$). To fuse these two context sources, they are aggregated into a single matrix to form an integrated context representation $\boldsymbol{H}_{\mathrm{sce}}$. Subsequently, an additional AM is employed:

$$\boldsymbol{h}_{\mathrm{endpt}} = \mathrm{Attn}(\boldsymbol{h}_{\mathrm{hist}}\boldsymbol{W}^{Q}_{\mathrm{sce}}, \boldsymbol{H}_{\mathrm{sce}}\boldsymbol{W}^{K}_{\mathrm{sce}}, \boldsymbol{H}_{\mathrm{sce}}\boldsymbol{W}^{V}_{\mathrm{sce}}), \tag{24}$$

where $\boldsymbol{W}^{Q}_{\mathrm{sce}}$, $\boldsymbol{W}^{K}_{\mathrm{sce}}$ and $\boldsymbol{W}^{V}_{\mathrm{sce}}$ are learnable parameters performing linear projection. The resulting $\boldsymbol{h}_{\mathrm{endpt}}$ can be interpreted as an endpoint encoding that integrates both map information and neighbouring interaction cues, with a hidden size of D_{goalAttn}.

By grounding goals in interpretable map elements, misguidance from over-reliance on single-lane centreline data can be avoided, and coarse learning from sets of less relevant scenario contexts can be prevented.

Finally, one FC layer is used to map the high-dimensional features to the goal point coordinates $\boldsymbol{P}_{\mathrm{goal}}$:

$$\boldsymbol{P}_{\mathrm{goal}} = \mathrm{FC}(\mathrm{cat}(\boldsymbol{h}^{0}_{\mathrm{hist}}, \boldsymbol{h}_{\mathrm{endpt}})). \tag{25}$$

Trajectory decoder

An LSTM decoder is used to generate the future trajectories of the TA. At any time t, the decoder generates the future trajectories over the subsequent f time steps. At each prediction time step, the inputs comprise $\boldsymbol{D}_0^{(0)}$ and $\boldsymbol{h}_{\text{endpt}}$, and the hidden and cell states of the LSTM in the decoder are initialised by the outputs of the trajectory encoder:

$$\boldsymbol{h}_{\text{pred}}^t, \boldsymbol{c}_{\text{pred}}^t = \text{LSTM}(\text{cat}(\boldsymbol{D}_0^{(0)}, \boldsymbol{h}_{\text{endpt}}), (\boldsymbol{h}_{\text{hist}}^0, \boldsymbol{c}_{\text{hist}}^0)) . \tag{26}$$

Then the MLP is used to output the deterministic coordinate at time t:

$$\hat{\boldsymbol{Y}}^{(t)} = \text{MLP}(\boldsymbol{h}_{\text{pred}}^t) . \tag{27}$$

Notably, the estimated goal point coordinate $\boldsymbol{P}_{\text{goal}}$ is excluded from the input of the LSTM decoder; instead, $\boldsymbol{h}_{\text{endpt}}$ is used for its richer contextual information compared to a 2D coordinate. However, $\boldsymbol{P}_{\text{goal}}$ is incorporated into the loss function to guide model training.

Training loss function

In the prediction structure, two intermediate outputs are generated: the most likely reachable map element and the potential goal point. These two outputs are highly related to the manoeuvre intention and an accuracy prediction improves the explanability of the prediction model. Thus, a comprehensive loss function is established to evaluate these two outputs, as well as the overall trajectory prediction accuracy.

The mean square error (MSE) is used to calculate the regression loss of trajectory points during training:

$$MSE = \frac{1}{p}\sum_{i=1}^{p}[(x_{t+i} - x_{t+i}^{\text{gt}})^2 + (y_{t+i} - y_{t+i}^{\text{gt}})^2] . \tag{28}$$

The goal point estimation loss is calculated as the deviation from the goal estimation to its ground truth:

$$SE_{\text{goal}} = [(x_{\text{goal}} - x_{t+p}^{\text{gt}})^2 + (y_{\text{goal}} - y_{t+p}^{\text{gt}})^2] . \tag{29}$$

The map element matching is a classification task. The negative log-likelihood (NLL) loss is used to calculate the matching loss:

$$NLL_{\text{map}} = -\log_2\left(\sum_{i=0}^{m} \boldsymbol{A}_{\text{attn}}^i \cdot \boldsymbol{Mask}_{\text{map_gt}}^i\right) , \tag{30}$$

where $\boldsymbol{Mask}_{\text{map_gt}}$ is an array of binary values and the ground truth is set to be one and others zero.

To address the spatial overlap issue of map elements in intersection and roundabout scenarios, a matching algorithm based on trajectory-complete lane similarity is used to accurately associate trajectory points with map elements by identifying the target driving lane. Since the prediction of bicycles does not rely on map elements, for a fair comparison, the NLL excludes the results of map element matching of bicycles.

The final loss function during training is as follows:

$$loss = \alpha\ MSE + \beta SE_{\text{goal}} + \gamma NLL_{\text{map}} , \tag{31}$$

where $\alpha = 1$, $\beta = 1$ and $\gamma = 1$ to match the scale of these three costs into the same order of magnitude.

Experiment evaluation

Numerical experiments have been conducted on the inD dataset[33] and the rounD dataset.[34] The inD dataset contains four unsignalised intersections and three of them (Bendplatz, Heckstrasse and Neukoellner) are used in this research, including one four-arm intersection and two T-junctions. One additional roundabout from the rounD dataset is selected to supplement samples with varying RLs and TA dynamics.

Data description

Agent data are split into training, validation and testing sets. To minimise the imbalance of the manoeuvre patterns (going through, turning left and turning right), data samples are split according to agent types and manoeuvre patterns. More specifically, with a selected agent type and a chosen manoeuvre, 80% of agent data are sliced into the training set, 10% to the validation set and 10% to the testing set. This configuration is similar to those used in works of Geng et al.[42,43] Detailed statistics of samples are shown in Table 3.

Evaluation metrics

Two widely used measures of prediction effectiveness are employed for performance evaluation. Lower values indicate better prediction performance.

Final Displacement Error (FDE)[44]: This metric calculates the L2 distance between the predicted trajectory endpoint and the ground truth endpoint:

$$FDE = \frac{\sum_{i=1}^{N}\sqrt{(x_{t+p} - x_{t+p}^{\text{gt}})^2 + (y_{t+p} - y_{t+p}^{\text{gt}})^2}}{N} , \tag{32}$$

where x_{t+p} and y_{t+p} represent the predicted 2D position coordinates of a TA at the last time step $t+f$ of the prediction horizon. The superscript gt represents that the data are from the ground truth; otherwise, from the prediction.

Table 3. Statistics of samples extracted from the inD and rounD datasets.

Agent Class	Mnvr	Intxn A			Intxn B			Intxn C			Intxn D		
		Trng	Valid	Tstg	Trng	Valid	Tstg	Trng	Valid	Tstg	Trng	Valid	Tstg
Cars	GTH	536	67	67	436	55	56	1106	138	138	72	9	9
	TL	372	47	48	87	11	12	136	17	17	17	3	2
	TR	310	39	40	80	10	11	178	22	23	21	3	2
Trucks	GTH	22	2	3	11	1	2	45	6	7	23	3	3
	TL	3	0	1	7	1	1	28	3	4	3	1	1
	TR	4	0	1	8	1	1	38	5	5	4	1	1
Bicycles	GTH	201	25	25	17	2	3	23	3	4	1	0	1
	TL	65	8	8	1	1	1	7	1	3	0	0	0
	TR	59	7	8	2	1	1	8	2	2	0	0	0
Overall		1572	195	201	649	83	88	1569	197	203	141	20	17

GTH: going through; TL: turning left; TR: turning right; trng: training; valid: validation; tstg: testing; Mnvr: manoeuvre; Intxn: intersection.

Table 4. Parameter values of the data pre-processing.

Param	Description	Value
m	The number of the map elements	600
m_{AOI}	The maximum number of map elements allowed in the AOI	40
N	A redundancy factor considering potential over-speeding in cBFS	1.8
$N_{\text{LC_max}}$	The maximum number of allowed lane changes	2
n	The maximum number of SAs	15
t_{pred}	Duration of the prediction horizon	4
t_{hist}	Duration of the observation horizon	2

The final trajectory prediction point is related to the TA's driving intention, so it is important to evaluate its accuracy.[44]

Average Displacement Error (ADE)[44]: It is used to measure the overall distance deviation of the predicted trajectory to the ground truth:

$$ADE = \frac{\sum_{i=1}^{N}\sum_{t=1}^{p}\sqrt{\left(x_{t+i} - x_{t+i}^{\text{gt}}\right)^2 + \left(y_{t+i} - y_{t+i}^{\text{gt}}\right)^2}}{N \cdot p}, \tag{33}$$

where x_{t+i}, y_{t+i} represent the predicted trajectory point at the timestep $t + i$.

Implementation details

In this research, the prediction model is aimed at performing trajectory prediction up to 4 s with an observation of the past 2 s, denoted as t_{pred} and t_{hist} respectively. The data in the inD are recorded at 25 Hz. For training purposes, these data are down-sampled to 5 Hz, resulting in a length of 11 in the historical data and a length of 20 in the prediction.

The model is implemented using PyTorch. In the model training, Adam is used as the optimiser, with a batch size of 128 and an initial learning rate of 0.001, respectively. The proposed method is trained for 40 epochs, and the learning rate is multiplied by 0.7 every 8 epochs. To reduce the overfitting, the dropout rate is set to be 0.3. The training is deployed on the Nvidia GPU A100 within a High-Performance-Computing (HPC) platform.

The general settings and the dimensions of layers in key modules are represented in Tables 4 and 5.

Results of cBFS

The cBFS produces a set of potential destination map elements and the top-K hit rates, which indicate positional identification of ground truth map elements within the top K results, are used for performance evaluation. The higher the hit rate is, the better the performance of exploration. Besides, missing rates (MR) are used for evaluation, which stand for the proportion of destination map elements unidentified in the search results. Alongside the proposed cBFS, DFS and intuitive range searching (RngS) are evaluated, given that they are widely used in applications.[8,29,45] In the performance analysis, the maximum number of stored map elements is set to 90 to show the complete searching results of the algorithms, while it is set to 40 in the DL application for lightweight storage. The cBFS and DFS search reachable map elements along the RL while the RngS collects the map elements within a radius in a homogeneous way.

In equation (7), a redundancy factor N should be pre-defined prior to conducting cBFS. To determine a reasonable value for N, kernel density estimation

Table 5. Parameter values in GoPred.

Param	Description	Value
Input dimension	Dimension of LSTM input in the trajectory encoder	4
D_{LSTM}	Dimension of LSTM hidden state	64
D_{nbrAttn}	Dimension of AM in neighbour interaction	64
D_{pos}	The dimension of coordinates	2
n_{cpt}	The number of centre points in a single map element	18
$D_{\text{conv_i}\in\{1,2,3\}}$	dimension of 1D Conv	64
D_{mapAttn}	Dimension of AM in map element selection	64
D_{goalAttn}	Dimension of AM in goal estimation	64

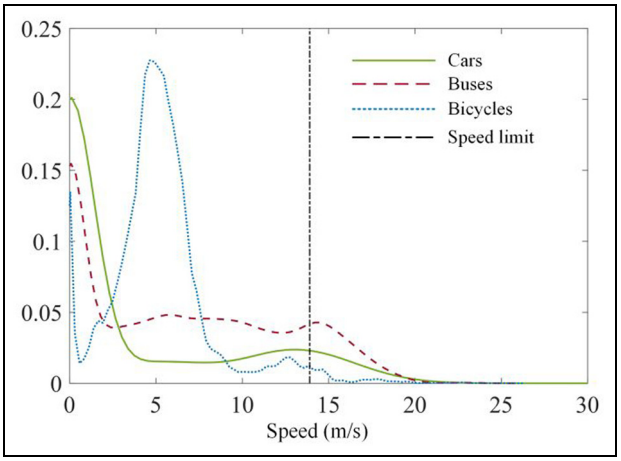

Figure 4. The speed distribution of cars, trucks and bicycles. The speed limit of the scenarios is 13.9 m/s and is marked as a black dash-dotted line. For all three traffic agents, there is a small peak near the speed limit, indicating that traffic agents tend to move around the speed limit. Besides, over-speeding cannot be ignored.

Table 6. Simulation results of vehicle dynamics under multiple lane changes.

Number of lane changes	Speed (km/h)	Extreme values		Final states at 4 s	
		a_{lat} (m/s^2)	ω (°/s)	a_{lat} (m/s^2)	ω (°/s)
1	20	0.082	8.080	0.051	6.771
	30	0.110	7.372	0.043	4.466
	40	0.123	6.247	0.033	2.787
	50	0.126	5.160	0.031	1.871
	60	0.130	4.371	0.028	1.196
2	20	0.181	17.967	0.156	17.534
	30	0.222	14.233	0.111	10.350
	40	0.248	12.252	0.089	6.663
	50	0.255	10.207	0.080	3.394
	60	0.259	8.682	0.069	2.789
3	20	0.252	23.394	0.277	23.078
	30	0.316	20.275	0.230	18.217
	40	0.361	17.885	0.194	12.416
	50	0.377	15.150	0.166	8.254
	60	0.384	12.971	0.136	5.236

(KDE) was performed to analyse the speed distributions of cars, buses, and bicycles. The resulting speed profiles are presented in Figure 4. To ensure coverage of 99.99% of real-world cases, N in equation (7) is set to be 1.8.

The second hyperparameter for cBFS is the maximum number of allowed lane changes, denoted as $N_{\text{LC_max}}$. Performing multiple lane changes within a short timeframe, for example, 4 s, can degrade ride comfort, increase driver workload and lead to a manoeuvre deviating from normal driving conditions.

To investigate the feasibility of such manoeuvres within the prediction horizon, simulations were conducted in CarSim. The speed range is set to 20 to 60 km/h, aligning with urban driving scenarios. Within a 4 s trajectory prediction horizon, a single lane change involves a lateral displacement of 3.5 m, a distance matching the typical width of driving lanes as referenced in literature.[46] Individual lane change paths are generated using quintic polynomial curves to ensure smooth transitions between lane centres during constant velocity travel. Results for lateral acceleration a_{lat} and yaw rate ω are summarised in Table 6.

Lateral acceleration and yaw rate are key indicators of ride comfort and manoeuvre stability. Simulation results reveal critical dynamics limitations associated with increasing lane change number within 4 s. At low speeds ($\leqslant$ 30 km/h), 2 to 3 consecutive lane changes produce yaw rates exceeding 15 °/s, such as 17.967 °/s for 2 lane changes at 20 km/h and 20.275 °/s for 3 lane changes at 30 km/h. These yaw rate values far surpass the 8 °/s threshold observed in naturalistic driving with an average value of 1.4 °/s,[47] indicating aggressive manoeuvring inconsistent with human driving norms. At higher speeds ($\geqslant$ 50 km/h), lateral acceleration becomes the constraining factor. For instance, the 3-lane-change mobility at 50 km/h generates peak lateral acceleration of 0.377 g and large yaw rates of 15.15 °/s, approaching the 0.4 g comfort limit defined by Bosetti et al.[47] and surpassing the naturalistic driving threshold of 8 °/s. Notably, final states at 4 s show non-zero yaw rates and lateral accelerations for 3-lane changes, for example,

Table 7. Statistics of the lane-changing number.

Number of lane changes	Number of agents
0	5602
1	180
2	21
>2	2

8.25 °/s and 0.166 g at 50 km/h, confirming incomplete manoeuvre execution within the defined prediction horizon. In contrast, two-lane changes maintained lateral acceleration below 0.3 g across all speeds and had final states with smaller residual dynamics.

Theoretical justification for the two-lane change constraint is further reinforced by scenario behavioural statistics. In the analysis, a lane change manoeuvre is defined as the TA transitions to either a direct neighbour of its current lane segment or a successor of the direct neighbour of its current lane segment, as shown in Figure 5.

Note that there are edge cases that travel across the road to perform a parking manoeuvre. They perform more than two lane changes within the prediction horizon. These cases are excluded from the dataset, as they fall outside the scope of the target prediction manoeuvres. The statistics of lane-changing numbers are provided in Table 7.

Based on the analysis of lane-changing manoeuvres of cars and trucks in the inD dataset and the simulation results based on Carism, $N_{\text{LC_max}}$ is set to be 2, considering the feasible number of lane changes within the prediction horizon of 4 s.

The results of the top 40 hit rates are shown in Table 8. Across all scenarios, cBFS demonstrates balanced performance characteristics. In Intxn A, cBFS achieves a hit rate reaching 0.996 at the top 40 elements with the minimal MR of 0.003, outperforming both DFS and RngS. DFS exhibits strong initial performance with 0.926 at the top 10 elements, but plateaus at 0.928. This is because DFS searches with a deep depth before switching to another direction. Though this is aligned with the truth that vehicles keep going straight most of the time due to the simplicity of the intersection arm topology, it fails to identify destination map elements promptly due to lane changing. In Intxn B, cBFS maintains 0.998 hit rates at the top 40 elements with a negligible MR of 0.001. In a more complex scenario, like Intxn C, where more lanes exist in one driving direction, LC actions exist with a higher possibility. This scenario highlights cBFS's performance, achieving a top 40 hit rate of 0.965 while DFS only achieves 0.922. Intxn D further demonstrates cBFS adaptability to road topology variation like roundabouts, where it reaches 0.996 hit rates by the top 20 elements, outperforming DFS delayed convergence. In all four scenarios, the performance of RngS varies a lot, with top 40 hit rates ranging from 0.439 to 0.871. This is because it cannot extend along the lane direction and includes too many map elements in the lane traversal directions, demonstrating poor adaptability to road layout variance.

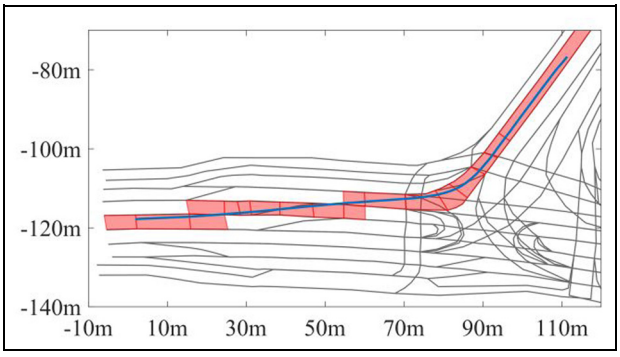

Figure 5. An example of an agent executing two lane changes in Intxn C. The trajectory direction is from right to left.

The detailed distributions of ranking hit rates at Intxn C and Intxn D are illustrated in Figure 6(a) and (b), respectively. The results reveal that DFS identifies ground truth map elements earlier in rankings but

Table 8. The results of the top-K hit rate.

		Top-K Hit Rate				MR (Top 90)
		5	10	20	40	
Intxn A	RngS	0.462	0.605	0.691	0.743	0.070
	DFS	0.823	0.926	0.927	0.928	0.050
	cBFS	**0.867**	**0.933**	**0.983**	**0.996**	**0.003**
Intxn B	RngS	0.504	0.594	0.656	0.732	0.142
	DFS	0.679	0.780	0.915	0.930	**0.001**
	cBFS	**0.684**	**0.785**	**0.921**	**0.998**	0.001
Intxn C	RngS	0.213	0.291	0.356	0.439	0.372
	DFS	0.438	**0.692**	0.788	0.922	0.033
	cBFS	**0.481**	0.686	**0.855**	**0.965**	**0.022**
Intxn D	RngS	0.217	0.274	0.510	0.871	**0**
	DFS	**0.598**	0.771	0.782	0.996	0.004
	cBFS	0.492	**0.885**	**0.996**	**0.996**	0.004

The bold means the best performance in comparison.

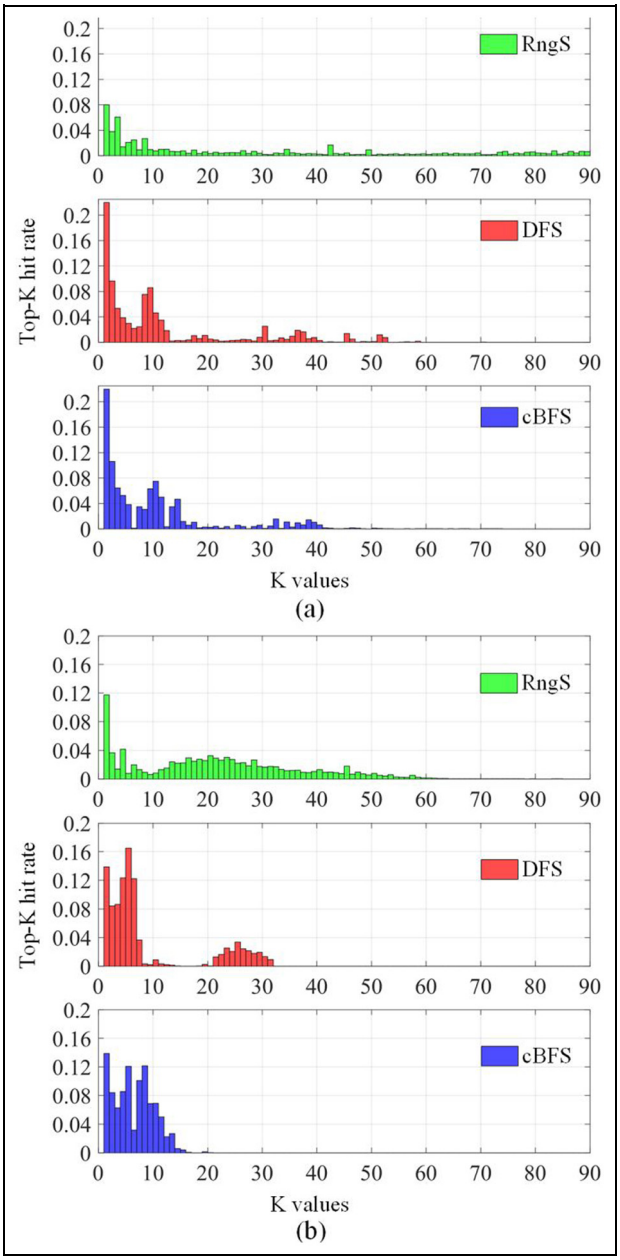

Figure 6. The top 90 ranking hit rate distributions at Intxn C and D are illustrated in (a) and (b), respectively.

struggles with topological variations. As Figure 6(b) shows, DFS's unidirectional deep search fails to detect the ground truth destination in early stages in the multi-entrance and multi-exit roundabout, with a secondary hit rate peak around position 25. In comparison, through searching on the longitudinal and lateral directions consecutively, cBFS can successfully identify ground truth during forward exploration and produces a more fluent ranking hit rate distribution. Additionally, RngS consistently shows dispersed hit rate distributions with delayed peaks, confirming its unoriented search mechanism inefficiency.

In terms of time efficiency, cBFS, DFS, and RngS require 113.0 s, 106.6 s, and 430.6 s, respectively, to complete AOI exploration across all four scenarios on a device equipped with an Intel Core i7-8700K CPU and 24 GB of memory. cBFS is efficient and exhibits performance comparable to DFS. Although RngS is theoretically efficient due to its unidirectional search mechanism, it performs 3,880,200 operations, substantially more than the 73,216 operations of both cBFS and DFS. The large number of operations for RngS arises because it searches the AOI trajectory on a point-by-point basis, whereas cBFS and DFS perform exploration processing only when occupied map elements change between consecutive trajectory points.

Trajectory prediction results

To validate the efficacy of the proposed GoPred, it is compared to some representative models based on the data described in Section 'Data description'. These include the vanilla LSTM, mCS-LSTM and GA-LSTM. The vanilla LSTM method uses an encoder-decoder structure, which is widely used and reported.[49,50] The encoder and decoder are set to be the same as those in Sections 'Trajectory encoder' and 'Trajectory Decoder' and they rely solely on TA's historical status. mCS-LSTM is modified from CS-LSTM which encodes the neighbour interaction through convolutional social pooling (CSP).[2] The manoeuvre probability module is eliminated from CS-LSTM since unimodal prediction is deployed in this work. GA-LSTM is a variation of the proposed method that shares most of the configuration. The neighbour interaction is encoded by AM, the same as GoPred, and the encoding of the estimated goal point is used to supplement the input of the decoder. The difference is that the interaction between the map and the TA is mined by a GCN. This GCN-based map mining module is inspired by the lane convolution operator laneGCN[3] and a stack of multi-scale GCNs are used. Specifically,

Table 9. Prediction results.

	#Param	Latency (ms)	GFLOPs	ADE				FDE			
				car	bus	bicycle	overall	car	bus	bicycle	overall
Vanilla LSTM	**35,458**	0.55	**0.009**	1.035	1.044	0.824	1.007	2.794	2.781	2.198	2.712
mCS-LSTM	114,962	1.43	0.024	0.890	0.906	0.723	0.868	2.296	2.264	1.846	2.233
GA-LSTM	159,748	25.11	0.042	0.817	0.838	0.624	0.792	2.203	2.192	1.580	2.117
GoPred	146,954	9.93	0.099	**0.761**	**0.803**	**0.611**	**0.743**	**2.011**	**2.047**	**1.495**	**1.943**

The bold means the best performance in comparison, such as the least number of parameters, the smallest GFLOPs and the smallest errors (ADE and FDE).

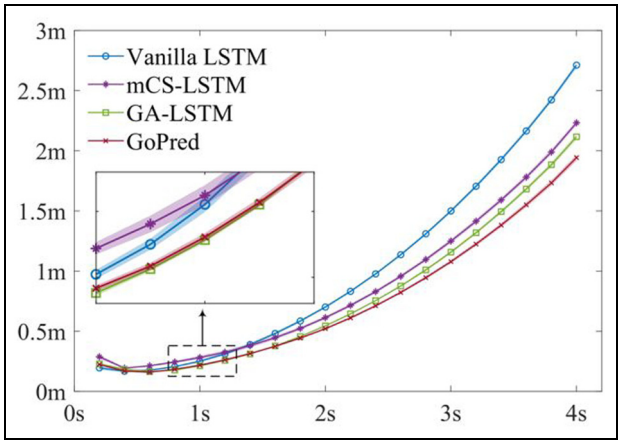

Figure 7. The DEs across the prediction accuracy. They increase with the growth of the prediction horizon. The minimum DE is not achieved at the closest prediction horizon (0.2 s) but at the second (0.4 s).

aggregation steps of 1, 2, 4 and 8 are deployed respectively with the same configuration of GCN and the results of these individual aggregations are fused to generate the final map-aware encoding and to supplement the input of the goal point estimation. This multi-scale GCN is used to reach topologically far-side map elements.

To ensure fairness in comparison, the hidden sizes of all models are kept consistent. Refer to Table 5 for details. Each model is trained 10 times and the average performances are shown in Table 9. Metrics of ADE and FDE are calculated, alongside the number of learnable parameters (#Param), inference latency and giga floating-point operations (GFLOPs).

Overall, the proposed GoPred achieves better performance than the others in terms of ADE and FDE. Specifically, when information from the SAs and RL is fused, the ADE and FDE are reduced by 26.2% and 28.4% respectively, compared to the baseline vanilla LSTM.

As shown in the comparison of mCS-LSTM and vanilla LSTM, neighbour interaction can improve the prediction accuracy. Improvements of 13.8% and 17.7% on ADE and FDE are reported in the comparison. Further prediction error reduction is achieved by supplementing RL information, as shown in the results of GA-LSTM and GoPred. By comparison, the AM is better at mining the relation between the TA and RL information. The ADE and FDE are reduced by 6.2% and 8.2% when AM is used compared to GCN in GA-LSTM.

Figure 7 illustrates displacement error (DE) evolution over the prediction horizon. As expected, prediction errors exhibit a general upward trend with increasing horizon length. Notably, GA-LSTM outperforms GoPred marginally for prediction horizons under 2 seconds. This discrepancy can be attributed to their architectural differences. GCN-based RL mining strategy in GA-LSTM prioritises local map element interactions, for example, immediate predecessors and neighbours. Though the multi-step message propagation strategy can reach topologically distant lane segments, their features become blurred during aggregation. Feature degradation from distant nodes in GCN is inevitable, which makes GCN focus more on local RL information and thereby provide better prediction performance in the short horizon as TA typically does not travel far. In contrast, the focus of GoPred is on potential destination inference and therefore provides better long-horizon performance.

An observation is that the minimum DE does not occur at the shortest horizon (0.2 s), likely due to stop-and-go manoeuvres introducing noise in low-speed trajectory data, e.g., from inherent data drift in the original dataset. Despite such noise at low speeds, no additional filtering was applied, as predictions remained within acceptable accuracy thresholds.

GFLOPs were measured with a batch size of 16, mimicking the traffic scenario where 16 TAs are around the AV.[29] This metric evaluates the computational complexity. In terms of computational efficiency, the vanilla LSTM achieves optimal performance, owing to its highly compact architecture that excludes additional neighbour or map interaction inputs. This minimal design results in a parameter size of 35.458 K. Conversely, GoPred exhibits the highest GFLOPs, as high as 0.098, with its computational overhead primarily attributed to the map element feature encoder and the downstream most likely map element estimator. The convolutional layers introduce a large number of learnable parameters, significantly increasing operational complexity and resulting in a 0.057 GFLOPs increment compared to GA-LSTM. Although GA-LSTM also incorporates RL processing, its computational demands remain moderate. Its GCN module processes RL data through MLPs and graph convolution operations, both of which feature low inherent complexity. While GA-LSTM demonstrates greater efficiency than GoPred, it exhibits limited long-term prediction performance.

Though having a higher complexity than other methods in this research, GoPred is still a lightweight model, compared to other dedicated models which normally have a parameter size larger than 200,000 and GFLOPs over 0.4.[29]

The inference latency of various models was evaluated on a single RTX 6000 GPU, with results presented in Table 9. Vanilla LSTM and mCS-LSTM demonstrate minimal latency at 0.55 ms and 1.43 ms, consistent with their low computational complexity and compact architecture. Notably, GoPred exhibits a latency of 9.93 ms despite its higher computational load of 0.099 GFLOPs. In contrast, GA-LSTM shows substantially elevated latency of 25.11 ms even with a lower theoretical complexity of 0.042 GFLOPs. This 15.18 ms difference between the two models presents a

Table 10. Per-rank hit rates of map element estimation.

First	Second	Third	Fourth	Fifth	top_5	w/ nbr
0.731	0.156	0.057	0.020	0.009	0.972	0.901

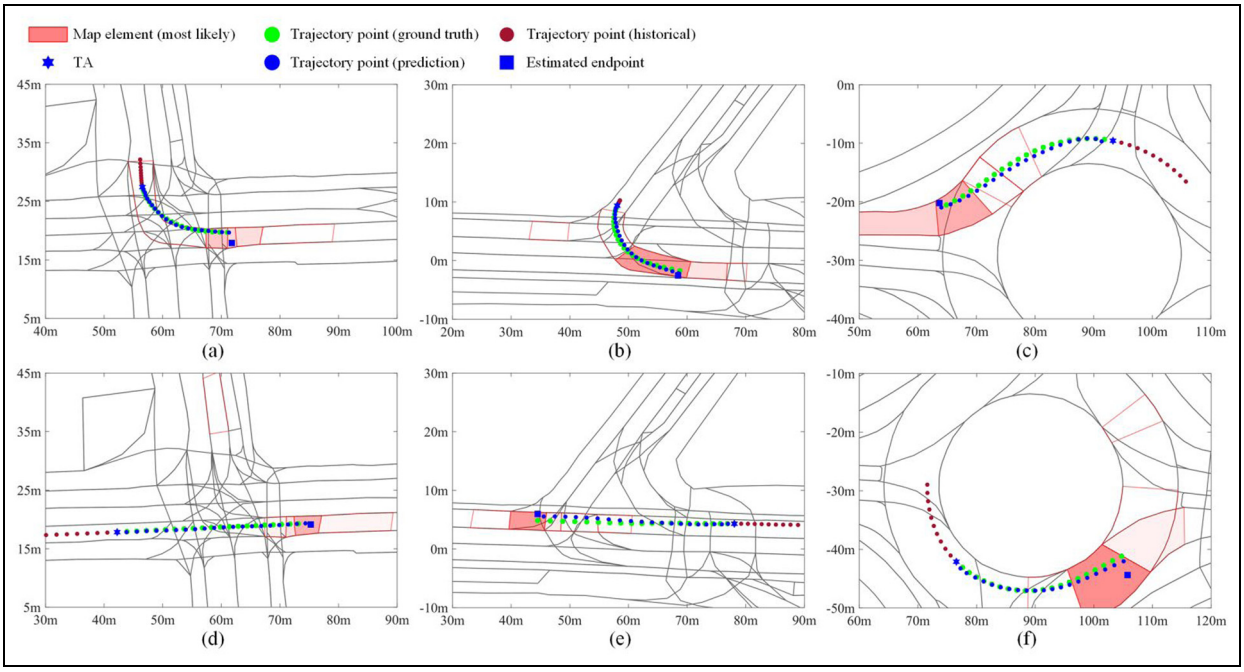

Figure 8. The prediction results of the proposed method. The top five most likely map elements are plotted in red with transparency. The higher the opacity, the higher the possibility. The estimated goal points are plotted as blue square markers. The waypoints of the historical trajectory, predicted trajectory and ground truth are represented by red, blue and green, respectively.

discrepancy that arises from the GCN-based map feature extraction module in GA-LSTM. This multi-layer component employs sparse matrix storage for graph adjacency representations, introducing inherent computational inefficiencies on GPU architectures. The sequential neighbourhood aggregation in GCN layers further increase latency by limiting parallelisation. In contrast, GoPred utilises compact storage by arranging key map elements within the AOI, minimising computations through focusing relevant map elements and executing efficient matrix operations.

The evaluation of map element estimation accuracy hinges on two metrics: per-rank hit rate and top-K (K = 5) hit rate. These metrics quantify the likelihood of the ground truth appearing at specific rank positions (first, second, …, fifth) and the probability that the ground truth is included in the top five predicted ranks, respectively.

As presented in Table 10, the estimation process successfully places the ground truth at the first rank with a 73.1% success rate. This result underscores a substantial degree of accuracy in initial guesses.

Though there is a notable decline in confidence from the first to the second rank, the top-5 hit rate remains high at 97.2%, indicating that a large majority of the ground truth elements are captured within the first five predictions. This reflects a broad enclosure of potential correct predictions.

An additional metric assesses the probability that the estimation aligns with either the first prediction or the direct neighbours of the first prediction, including the immediate successor, predecessor, right and left neighbours. The result is represented in the last column in Table 10. The value reaches 90.1%, indicating an accurate prediction. This metric is particularly critical for evaluating the map element estimation performance since it is vague to assign a deterministic map element to the TA while it is transitioning between two adjacent map elements.

To better demonstrate the prediction performance of the proposed method, the visualisation results of prediction at specific intersections are illustrated, as shown in Figure 8. As illustrated, the most likely map element estimator effectively identifies the most relevant map elements inside AOI. When the TA approaches the intersection, the estimator generates confidence scores for associated map elements, quantifying uncertainties in moving intentions. Notably, TA reaches the most probable map elements at the end of the prediction horizon, though not exactly, validating the module's

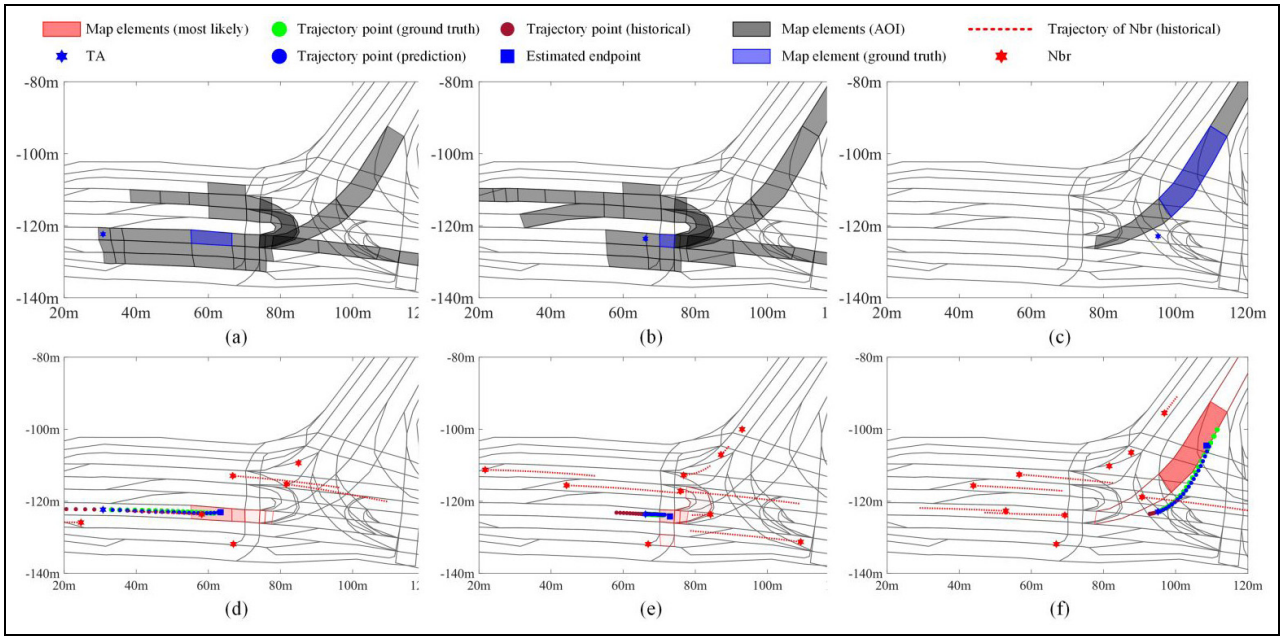

Figure 9. An example of TA executing a left-turn manoeuvre at the intersection. The top row demonstrates map elements inside AOI as grey polygons, with ground truth map elements highlighted in blue to represent the actual intention of the selected sample. The bottom row illustrates the trajectory prediction results. The waypoints of the historical trajectory, predicted trajectory and ground truth are represented by red, blue and green, respectively. Meanwhile, SAs are marked as red hexagons alongside their historical positions, providing context for the dynamic traffic environment around the TA.

effectiveness. Furthermore, the goal point prediction is softly constrained by the most likely map element and shows alignment with the ground truth endpoint. Both the map element estimation and goal point prediction progressively reveal the TA's movement intentions, facilitating enhanced interpretability of the prediction model. The consistency between map element estimation and goal prediction further confirms the model's effectiveness in narrowing the inference space and inferring movement intentions.

To provide a detailed illustration of how the prediction model estimates TA intention within intersections, a left turn case is selected, as shown in Figure 9. As the TA approaches the intersection, cBFS first identifies reachable map elements (Figure 9(a)), with initial most-likely map element predictions indicating a lane-keeping manoeuvre (Figure 9(d)). Upon nearing the intersection entrance, the TA decelerates to prepare for turning while avoiding collisions with SAs. At this stage, the estimated map elements suggest three potential manoeuvres: left U-turn, continuation to a proximal map element on the current lane or right lane change. Due to reduced speed, close side map elements are paid attention to.

As the TA enters the intersection, cBFS dynamically narrows the reachable map elements to those on the left-turn lane. Consequently, a high-confidence score is assigned to the left-turn lane map element, establishing a coarse-grained destination. Across all three sub-scenarios in Figure 9, corresponding goal points are predicted in alignment with the most likely map elements. These goal points not only align spatially with their associated map elements but also provide finer-grained localisation of the TA's intention, particularly valuable given the extended length of map elements.

Ablation experiments

This section investigates the significance of critical components in trajectory prediction models. GoPred progressively narrows the search space by exploring AOI and scoring corresponding map elements. It then generates a comprehensive goal encoding by integrating AOI encoding and neighbourhood interaction encoding, which further facilitates the prediction.

In GoPred, neighbour interaction encoding plays a vital role in estimating the most likely map elements and goal points. The estimated map elements help constrain potential goal generation and contribute to forming informative goal encodings. To further examine their impact on trajectory prediction, ablation experiments were conducted by removing relevant modules from the prediction backbone. Performance was evaluated using map element estimation accuracy, goal estimation deviation, ADE and FDE.

The ablation results show that the model with all modules in position achieved optimal ADE and FDE performance, as it simultaneously incorporates map information and neighbour interactions, as shown in Test 1 of Table 11. When map information was excluded, as shown in Test 2, the model relied on supplementary neighbour interactions to estimate potential goals. This omission increased goal prediction errors, leading to 13.7% and 12.3% rises in ADE and FDE,

Table 11. Results of ablation experiments.

	Nbr Enc	Map Enc	End Enc	Map Elem Pred Accu			Goal Devi	ADE	FDE
				First	top_5	w/nbr			
1	✓	✓	✓	**0.731**	**0.972**	**0.901**	**2.959**	**0.743**	**1.943**
2	✓	×	✓	-	-	-	3.607	0.845	2.182
3	×	✓	✓	0.709	0.967	0.886	3.781	0.864	2.317
4	✓	✓	×	0.691	0.959	0.878	-	0.807	2.086

The bold means the best performance in comparison tests.

Table 12. Results while different loss functions are used.

	Goal loss	Map matching loss	MSE	Map Elem Pred Accu			Goal Devi	ADE	FDE
				First	top_5	w/nbr			
1	✓	✓	✓	**0.731**	**0.972**	**0.901**	**2.959**	**0.743**	**1.943**
2	×	✓	✓	0.728	0.972	0.899	21.133	0.764	2.006
3	✓	×	✓	0.073	0.334	0.182	3.185	0.795	2.068
4	×	×	✓	0.005	0.123	0.050	21.130	0.796	2.099

The bold means the best performance in comparison tests.

respectively. Excluding neighbour interactions caused the largest performance degradation. As shown in Test 3, first rank hit rate dropped by 3.1%, while ADE and FDE surged by 16.3% and 19.2%. This highlights strong interactions between TAs and SAs at urban intersections, which significantly influence decision-making and motion. In Test 4, the goal encoding was replaced by the map element encoding in the trajectory decoder input. The results showed that map element encoding can facilitate the trajectory prediction, while the comprehensive goal encoding provides better guidance for trajectory generation.

GoPred progressively narrows down the search space. The TA's moving intention can be explicitly illustrated by the most likely map element that the TA will reach and the estimated goal point. Accurate estimation of these two parts relies on specific constraints in the training process. In Table 12, the results of map element matching accuracy and goal deviation from the estimation to the ground truth, as well as ADE and FDE, are shown when different combinations of loss functions are used in the model training.

In Test 3, it is obvious that excluding the map element matching loss leads to a remarkable increase in map element estimation error, the matching accuracy of which is as low as 7.3%. The ADE and FDE increase by 2.8% and 6.4%, respectively. Though the goal estimator still functions, it uses encodings of less relevant map elements, thereby degrading the performance. In Test 2, the goal point estimation loss is omitted. The model still focuses on proper map elements within the AOI. However, without the goal estimation loss, the encodings of map elements and neighbour interactions are not fused optimally, leading to a decrease in the prediction accuracy compared to the model that employs all three loss components. In Test 4, only MSE loss is used during training. Though the model still achieves acceptable prediction accuracy by learning from dynamic scene context, the performance degrades and the interpretability declines because the motion intention is not clear, with first rank hit rate being as low as 0.5% and the goal deviation being as large as 21.13 m.

Conclusions

The research presents a goal-oriented trajectory prediction framework for urban intersections, characterised by low latency and lightweight architecture. Leveraging fragmented map elements, the model enhances prediction accuracy and interpretability by progressively narrowing the inference space and focusing on the most influential map elements. Experimental results have shown that:

(1) By incorporating behavioural constraints, the proposed cBFS can uniformly explore AOI in both longitudinal and lateral directions and reflect real-world driving dynamics. It can provide robust performance across diverse road scenarios, handle topological variations, achieve minimal missing rates and maintain computational efficiency.
(2) The prediction model accurately identifies map elements that the TA is likely to reach, enhancing prediction accuracy.

Prediction interpretability is improved by estimating agents' intentions through the most relevant map elements and grounding corresponding goal points based on this information. This approach provides valuable

insights into the model's understanding of movement intentions and subsequent destinations.

The flexibility of cBFS in setting search limitations along both longitudinal and lateral lanes allows adaptation to diverse application scenarios. The proposed goal-oriented prediction model can fuse both RL and SA information, making it suitable for various scenes and agent classes.

To further validate the framework's generalisation ability, future work will augment it by incorporating additional scenarios through data merging with datasets such as highD and Argoverse. This will introduce variability in scenarios, enabling further exploration of adaptive capabilities.

Additionally, the proposed prediction model can be extended to better account for prediction uncertainties. Accurate trajectory predictions facilitate risk assessment and trajectory planning, while uncertainty-aware predictions better accommodate ambiguous intentions and perceptual noise.

A straightforward refinement of the proposed framework to address uncertainty is training the model to output trajectories with a Gaussian distribution. Additionally, the estimated top K most likely map elements indicate driving intentions and can serve as initial prediction proposals. Multiple trajectories may thus be generated from these probabilistic intermediate outputs, enabling multi-modal prediction. This approach mitigates the risk of uni-modal predictions learning average manoeuvres across diverse driving intentions.

Uncertainty-aware trajectory prediction supports obstacle avoidance in trajectory planning. However, handling the uncertainty is still an open challenge. In multi-modal prediction, the prediction with the highest probability is not guaranteed to have the least prediction errors. Adopting predictions from all modals often results in overly conservative plans due to overreacting to low-probability branches, while relying solely on the most probable modal can lead to collisions due to overconfidence. Consequently, the practical implications of prediction outcomes in planning tasks require further exploration.

ORCID iD

Marco Cecotti https://orcid.org/0000-0002-4457-3631

Funding

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the China Scholarship Council under Grant No.202108690001 and the Graduate Research and Innovation Projects of Jiangsu Province under Grant KYCX21_3334. We would like to express our sincere gratitude to Harikrishnan Vijayakumar for his suggestions on the model improvement.

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

1. Yin C, Cecotti M, Auger DJ, et al. Deep-learning-based vehicle trajectory prediction: A review. *IET Intell Transp Syst* 2025; 19(1): e70001
2. Deo N and Trivedi MM. Convolutional social pooling for vehicle trajectory prediction. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Salt Lake City, UT, USA, 18–22 June 2018, pp. 1468–1476. New York: IEEE.
3. Liang M, Yang B, Hu R, et al. Learning Lane Graph Representations for Motion Forecasting. In: *European Conference on Computer Vision*, Glasgow, UK, 23–28 August 2020, pp. 541–556. Cham: Springer International Publishing.
4. Meng Z, He R, Wu J, et al. GIVA: Interaction-aware trajectory prediction based on GRU-Improved VGG-Attention Mechanism model for autonomous vehicles. *Proc Inst Mech Eng Part D J Automob Eng* 2023; 239(1): 128–141.
5. Zhang K, Zhao L, Dong C, et al. AI-TP: Attention-based interaction-aware trajectory prediction for autonomous driving. *IEEE Trans Intell Transp Syst* 2023; 8(1): 73–83.
6. Yang Z, Li D, Wang Z, et al. Vehicle trajectory prediction based on attention optimized with real-scene sampling. *Syst Sci Control Eng* 2024; 12(1): 2347889.
7. Zeng W, Liang M, Liao R, et al. LaneRCNN: distributed representations for graph-centric motion forecasting. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic, 27 September–1 October 2021, pp. 532–539. New York: IEEE.
8. Deo N, Wolff E and Beijbom O. Multimodal trajectory prediction conditioned on lane-graph traversals. In: *Proceedings of the 5th Conference on Robot Learning*, London, UK, 08–11 November 2022, pp. 203–212. PMLR
9. Xin L, Wang P, Chan CY, et al. Intention-aware Long Horizon Trajectory Prediction of Surrounding Vehicles using Dual LSTM Networks. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, USA, 4–7 November 2018, pp. 1441–1446. New York: IEEE.
10. Messaoud K, Deo N, Trivedi MM, et al. Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. In: *2021 IEEE Intelligent Vehicles Symposium (IV)*, Nagoya, Japan, 11–17 July 2021, pp. 165–170. New York: IEEE.
11. Cui H, Radosavljevic V, Chou FC, et al. Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks. In: *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, 20–24 May 2019, pp. 2090–2096. New York: IEEE.
12. Wei L, Yang H, Niu Y, et al. Wheat biomass, yield, and straw-grain ratio estimation from multi-temporal UAV-based RGB and multispectral images. *Biosyst Eng* 2023; 234(1): 187–205.

13. Zhang Z, Yang M, Pan Q, et al. Identification of tea plant cultivars based on canopy images using deep learning methods. *Sci Hortic* 2025; 339(1): 113908.
14. Ren Y, Lan Z, Liu L, et al. EMSIN: enhanced multi-stream interaction network for vehicle trajectory prediction. *IEEE Trans Fuzzy Syst* 2024; 33(1): 54–68.
15. Gao J, Sun C, Zhao H, et al. VectorNet: encoding HD maps and agent dynamics from vectorized representation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 13–19 June 2020, pp. 11525–11533. New York: IEEE.
16. Tao K, Wang A, Shen Y, et al. Peach flower density detection based on an improved CNN incorporating attention mechanism and multi-scale feature fusion. *Horticulturae* 2022; 8(10): 904.
17. Kim B, Park S, Lee S, et al. LaPred: lane-aware prediction of multi-modal future trajectories of dynamic agents. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, 20–25 June 2021, pp. 14636–14645. New York: IEEE.
18. Bhattacharyya P, Huang C and Czarnecki K. SSL-lanes: self-supervised learning for motion forecasting in autonomous driving. arXiv preprint arXiv: 2206.14116, 2022.
19. Zhang L, Li P, Chen J, et al. Trajectory prediction with graph-based dual-scale context fusion. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan, 23–27 October 2022, pp. 3708–3715. New York: IEEE.
20. Guo H, Meng Q, Zhao X, et al. Map-enhanced generative adversarial trajectory prediction method for automated vehicles. *Inf Sci* 2023; 622(1): 1033–1049.
21. Pan J, Sun H, Xu K, et al. Lane-Attention: Predicting Vehicles' Moving Trajectories by Learning Their Attention Over Lanes. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 25–29 October 2020, pp. 7949–7956. New York: IEEE.
22. Luo C, Sun L, Dabiri D, et al. Probabilistic multi-modal trajectory prediction with lane attention for autonomous vehicles. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 24 October 2020–24 January 2021, pp. 2370–2376. New York: IEEE.
23. Liu M, Cheng H, Chen L, et al. LAformer: trajectory prediction for autonomous driving with lane-aware scene constraints. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seattle, WA, USA, 17–18 June 2024, pp. 2039–2049. New York: IEEE.
24. Wang X, Su T, Da F, et al. ProphNet: efficient agent-centric motion forecasting with anchor-informed proposals. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada, 17–24 June 2023, pp. 21995–22003. New York: IEEE.
25. Gómez-Huélamo C, Conde MV, Barea R, et al. Efficient baselines for motion prediction in autonomous driving. *IEEE Trans Intell Transp Syst* 2024; 25(4): 4192–4205.
26. Gao X, Jia X, Li Y, et al. Dynamic scenario representation learning for motion forecasting with heterogeneous graph convolutional recurrent networks. *IEEE Robot Autom Lett* 2023; 8(5): 2946–2953.
27. Gu J, Sun C and Zhao H. DenseTNT: end-to-end trajectory prediction from dense goal sets. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada, 10–17 October 2021, pp. 15303–15312. New York: IEEE.
28. Gilles T, Sabatini S, Tsishkou D, et al. GOHOME: graph-oriented heatmap output for future motion estimation. In: *2022 IEEE International Conference on Robotics and Automation (ICRA)*, Philadelphia, PA, USA, 23–27 May 2022, pp. 9107–9114. New York: IEEE.
29. Li L, Wang X, Lian J, et al. Efficient vehicle trajectory prediction with goal lane segments and dual-stream cross attention. *IEEE Trans Intell Transp Syst* 2024; 25(12): 1–14.
30. Gan L, Li P, Li J, et al. Goal-based neural physics vehicle trajectory prediction model. *Transp Res Part C Emerg Technol* 2025; 179(1): 104923.
31. Aydemir G, Akan A and Güney F. Adapt: efficient multi–agent trajectory prediction with adaptation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Paris, France, 2–6 October 2023, pp. 8965–8975. New York: IEEE.
32. Poggenhans F, Pauls JH, Janosovits J, et al. Lanelet2: A high-definition map framework for the future of automated driving. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, USA, 4–7 November 2018, pp. 1672–1679. New York: IEEE.
33. Bock J, Krajewski R, Moers T, et al. The inD dataset: a drone dataset of naturalistic road user trajectories at German intersections. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*, Las Vegas, NV, USA, 19 October–13 November 2020, pp. 1929–1934. New York: IEEE.
34. Krajewski R, Moers T, Bock J, et al. The rounD Dataset: A Drone Dataset of Road User Trajectories at Roundabouts in Germany. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, Rhodes, Greece, 20–23 September 2020, pp. 1–6. New York: IEEE.
35. Yin C, Cecotti M, Auger D, et al. Lane Centerline Extraction Based on Surveyed Boundaries: An Efficient Approach Using Maximal Disks. *Sensors* 2025; 25(8): 2571.
36. Zhu F, Chen J, Guan Z, et al. Development of a combined harvester navigation control system based on visual simultaneous localization and mapping–inertial guidance fusion. *J Agric Eng* 2024; 55(2): 22–45.
37. Feng S, Liu F, Zhou S, et al. Dynamic obstacle avoidance strategy for autonomous vehicles based on LSTM model trajectory prediction under urban roads. *Proc Inst Mech Eng Part D J Automob Eng* 2024; 239(10): 5275–5288.
38. Guan Y, Li N, Chen P, et al. Vehicle trajectory prediction method integrating spatiotemporal relationships with hybrid time-step scene interaction. *Proc Inst Mech Eng Part D: J Automob Eng* 2024; 239(10): 4666–4679.
39. Zuo X, Chu J, Shen J, et al. Multi-granularity feature aggregation with self-attention and spatial reasoning for fine-grained crop disease classification. *Agriculture* 2022; 12(9): 1499.
40. Wang Y, Li T, Chen T, et al. Cucumber downy mildew disease prediction using a CNN-LSTM approach. *Agriculture* 2024; 14(7): 1155.
41. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: *Advances in Neural Information Processing*

Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017, pp. 5998-6008. Red Hook: Curran Associates.
42. Geng M, Cai Z, Zhu Y, et al. Multimodal vehicular trajectory prediction with inverse reinforcement learning and risk aversion at urban unsignalized intersections. *IEEE Trans Intell Transp Syst* 2023; 24(11): 12227–12240.
43. Geng M, Chen Y, Xia Y, et al. Dynamic-learning spatial-temporal Transformer network for vehicular trajectory prediction at urban intersections. *Transp Res Part C Emerg Technol* 2023; 156(1): 104330.
44. Huang R, Zhuo G, Xiong L, et al. A review of deep learning-based vehicle motion prediction for autonomous driving. *Sustainability* 2023; 15(19): 14716.
45. Tian W, Wang S, Wang Z, et al. Multi-modal vehicle trajectory prediction by collaborative learning of lane orientation, vehicle interaction, and intention. *Sensors* 2022; 22(12): 4295.
46. Bae I, Kim J, Moon J, et al. Lane change Maneuver based on Bezier curve providing comfort experience for autonomous vehicle users. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand, 27–30 October 2019, pp. 2272–2277. New York: IEEE.
47. Huang X, Zhang S and Peng H. Developing robot driver etiquette based on naturalistic human driving behavior. *IEEE Trans Intell Transp Syst* 2020; 21(4): 1393–1403.
48. Bosetti P, Da Lio M and andSaroldi A. On the human control of vehicles: an experimental study of acceleration. *Eur Transp Res Rev* 2013; 6(2): 157–170.
49. Zyner A, Worrall S and Nebot E. Naturalistic driver intention and path prediction using recurrent neural networks. *IEEE Trans Intell Transp Syst* 2020; 21(4): 1584–1594.
50. Park S, Kim B, Kang C, et al. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China, 26–30 June 2018, pp. 1672–1678. New York: IEEE.

https://dspace.org/

Goal oriented trajectory prediction conditioned on reachable road context

Yin, Chenhui; Cecotti, Marco; Jiang, Haobin

2026-12-31
Sage
https://doi.org/10.1177/09544070251397454
Yin C, Cecotti M, Jiang H. (2025) Goal oriented trajectory prediction conditioned on reachable road context. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, Available online 15 December 2025

Downloaded from a DSpace repository. https://dspace.org/