

CRANFIELD UNIVERSITY

CHENHUI YIN

PREDICTION OF TRAFFIC AGENTS' TRAJECTORIES IN URBAN
SCENARIOS USING DEEP LEARNING

SCHOOL OF AEROSPACE, TRANSPORT AND
MANUFACTURING
Phd In Transport Systems

PhD

Academic Year: 2021 - 2024

Supervisor: Dr Marco Cecotti
Associate Supervisor: Dr Abbas Fotouhi, Prof Daniel Auger
June 2024

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND
MANUFACTURING
Phd In Transport Systems

PhD

Academic Year 2021 - 2024

CHENHUI YIN

Prediction of Traffic Agents' Trajectories in Urban Scenarios Using
Deep Learning

Supervisor: Dr Marco Cecotti
Associate Supervisor: Dr Abbas Fotouhi, Prof Daniel Augar
June 2024

This thesis is submitted in partial fulfilment of the requirements for
the degree of PhD

© Cranfield University 2024. All rights reserved. No part of this
publication may be reproduced without the written permission of the
copyright owner.

ABSTRACT

Trajectory prediction of surrounding traffic agents is essential for autonomous vehicles to identify and alleviate potential conflicts. Deep learning-based (DL-based) methods have gained attention because of their ability to process complex traffic scene contexts, like agent-to-agent and agent-to-road interaction. Extensive research has been conducted on trajectory prediction in highway scenarios, but limited studies focus on urban environments because of the complexity of interactions. This project selects a naturalistic driving dataset targeting urban intersections and trains DL-based models to predict the trajectories of high-mobility traffic agents. Traffic agents' track data and corresponding traffic scene data, like neighbour and road layout (RL) information, are extracted as training samples from the selected dataset. A novel method based on maximal disks is proposed to extract lane centrelines, which are important RL features but not available in most datasets. A trajectory prediction backbone is proposed based on an RNN-based encoder-decoder structure. Experiments showed that particular architectural decisions, like using LSTM instead of GRU, explicit calculation of the velocity of traffic agents, and close coupling between the encoder and the decoder, can improve the prediction accuracy. Further, the influences of agent interaction and road scene mining were studied separately. A model using Convolutional Social Pooling to extract agent interaction was replicated from the literature and was improved for better performance. In road scene mining, map element features are extracted from the sparsely defined RL map, and agent-map attention is used to extract road scene context. The models are trained in multiple scenarios with diverse RLs. Experimental evaluations have shown that both agent interaction and road scene mining can improve prediction accuracy. This project treats the agent interaction and road scene mining separately, future work includes exploring the fusion of these two encodings and more efficient neural networks in arranging traffic scene elements and extracting corresponding features.

Keywords:

neural network, centreline extraction, urban intersection, road scene

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisors, Dr. Marco Cecotti, Dr. Abbas Fotouhi, and Prof. Daniel Auger. Their unwavering support, both academically and personally, has been invaluable. Without their generosity in sharing their vast knowledge and expertise, I could not have undertaken this PhD journey. Their highest quality of guidance and the stress-free research environment they fostered allowed me to truly enjoy my work and pursue my research with enthusiasm.

I extend my heartfelt thanks to the China Scholarship Council for awarding me a scholarship under the State Scholarship Fund, enabling me to pursue my studies in the UK. This opportunity has been instrumental in my academic development. Additionally, I am deeply grateful to Prof. Haobin Jiang, my supervisor at Jiangsu University, for providing me with the opportunity to attend the double-degree doctoral program with Cranfield University. His support has been pivotal in broadening my academic horizons and enriching my research experience.

Finally, I am profoundly indebted to my parents for their unwavering support and belief in me. Their shared life experiences and constant encouragement have been my pillars of strength. Their support has kept my spirits high and my motivation strong throughout this PhD journey. Words cannot adequately express the depth of my gratitude for their sacrifices and unwavering faith in my abilities.

Thank you all for being an integral part of my journey.

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENTS.....	iii
LIST OF FIGURES.....	ix
LIST OF TABLES	xv
LIST OF EQUATIONS.....	16
LIST OF ABBREVIATIONS	19
1 Introduction.....	23
1.1 Background.....	23
1.1.1 Project Aim and Objectives	24
1.1.2 Thesis Outline	25
1.1.3 Novelty of This Project	28
2 Review on Naturalistic Driving Datasets.....	31
2.1 Naturalistic Driving Dataset.....	31
2.1.1 On-Board Sensors.....	31
2.1.2 Infrastructure-Fixed Sensors	35
2.1.3 Aerial Sensors.....	37
2.1.4 A Summary of Three Types of Recording Methods	40
2.2 Map Formats Describing Traffic Scenarios	42
2.2.1 OpenDrive	42
2.2.2 OpenStreetMap.....	43
2.2.3 Liblanelet.....	44
2.2.4 Lanelet2	46
2.2.5 A Summary of Map Formats	48
3 Deep-Learning-Based Vehicle Trajectory Prediction: A Review.....	49
3.1 Abstract.....	49
3.2 Introduction	49
3.3 Problem Formulation of Vehicle Trajectory Prediction	53
3.4 Fundamental Deep Learning Algorithms for Trajectory Prediction.....	55
3.5 Classification of DL-Based Trajectory Prediction Methods	57
3.5.1 Basic Neural Networks for Trajectory Prediction Backbone	58
3.5.2 Neural Networks for Interaction Mining	65
3.5.3 Neural Networks for Road Scene Mining	75
3.5.4 Neural Networks for Fusion.....	82
3.6 Performance Comparison	88
3.6.1 Evaluation Metric.....	88
3.6.2 Comparison of State-of-the-art Methods	89
3.6.3 Discussions of Future Research Directions.....	98
3.7 Conclusion	99
4 Framework Methodology	101
4.1 Dataset Comparison and Selection.....	103

4.1.1 Sufficient Data.....	104
4.1.2 High Position Accuracy	104
4.1.3 Comprehensive Interaction	105
4.2 Introduction of Intersection Drone Dataset (inD)	106
4.2.1 An Overall View.....	106
4.2.2 Separate Scenarios.....	108
4.3 Statistics Analysis of Selected Intersections	110
4.3.1 The Number of Road Users and Their Traversal Time.....	110
4.3.2 Analysis of Manoeuvres of Traffic Agents	112
4.4 Data Pre-processing	115
4.4.1 Used tools	115
4.4.2 Map Data Processing	117
4.4.3 Track Data Processing	122
4.4.4 Data Supplement	126
4.4.5 Dataset Splitting for DL Purpose	127
4.4.6 Standardisation	128
4.5 Trajectory Prediction Model Design Based on Kinematic Models.....	129
4.5.1 Introduction of Physical-based Models.....	129
4.5.2 Dealing with Uncertainties by Kalman Filters	130
4.5.3 Problem Statement.....	131
4.5.4 Applied Kinematic Models	132
4.5.5 Deployment of Kalman Filters with The Kinematic Models.....	134
4.5.6 Trajectory Prediction with The Kinematic Models.....	139
4.6 Trajectory Prediction Model Design Based on DL Models	140
4.6.1 Inputs and Outputs.....	140
4.6.2 Useful methods	142
4.6.3 Implementation Details.....	144
4.7 Evaluation Metrics.....	144
5 Parametrization of Road Scenes Based on Surveyed Boundaries.....	147
5.1 Abstract.....	147
5.2 Introduction	147
5.2.1 Related Work	148
5.2.2 Motivations and Contributions	152
5.3 Methodology	154
5.3.1 The Extraction of Lane Centre Points Based on MD	154
5.3.2 Segment Pairing.....	157
5.3.3 Circle Centres Filtering.....	159
5.3.4 Connectivity of Circle Centres	159
5.3.5 The Extraction of Lane Centrelines Based on VT	161
5.3.6 The Extraction of Lane Centrelines Based on DT	162
5.4 Experiments and Results	165
5.4.1 Data Description.....	165

5.4.2 Experimental Results and Comparisons	166
5.5 Conclusions	171
6 Trajectory Prediction Model based on LSTM and Agent Historical Status ..	173
6.1 Introduction	173
6.1.1 Related Research	173
6.1.2 Motivations and Contributions	174
6.2 Prediction Model	175
6.2.1 Trajectory Encoder	176
6.2.2 Trajectory Decoder	176
6.3 Experimental Evaluation	176
6.3.1 Compared Models	176
6.3.2 Trajectory Prediction Results	177
6.4 Conclusions and Future Work	185
7 Trajectory Prediction by Considering The Interactions between The Traffic Agents	187
7.1 Introduction	187
7.1.1 Related Research	187
7.1.2 Motivations and Contributions	187
7.2 Prediction Model	188
7.2.1 Trajectory Encoder	190
7.2.2 Convolutional Social Pooling	190
7.2.3 Trajectory Decoder	192
7.3 Experimental Evaluation	192
7.3.1 Compared Models	192
7.3.2 Trajectory Prediction Results	193
7.4 Conclusions and Future Work	195
8 Trajectory Prediction With Road Scene Mining	197
8.1 Introduction	197
8.1.1 Related Work	197
8.1.2 Motivations and Contributions	201
8.2 Prediction Model	202
8.2.1 Map Element Feature Encoder	202
8.2.2 Trajectory Encoder	204
8.2.3 Agent to Map Element Attention	204
8.3 Experimental Evaluation	205
8.3.1 Compared Models	206
8.3.2 Implementation Details	206
8.3.3 Trajectory Prediction Results	206
8.4 Conclusions and Future Work	211
9 Conclusions and Future Work	213
9.1 Conclusions	213
9.2 Discussion on Generalisation	216

9.3 Future Work.....	217
REFERENCES.....	219

LIST OF FIGURES

Figure 1-1 A diagram of relationships between the objectives and the chapters	27
Figure 2-1 The test car used in KITTI data collection (a) and the layout of sensors (b) [7]	32
Figure 2-2 The test car used to record data in Argoverse 1 [8]	33
Figure 2-3 The test car used in collecting ACFR [12]	34
Figure 2-4 The bird-eye-view of one recorded intersection, including the test vehicle (in red), and target intersection arms (blue arrows) [12].	35
Figure 2-5 An illustration of the infrastructure-fixed camera in NGSIM [14].	37
Figure 2-6 An illustration of scenarios that the Stanford drone dataset covered [17].	38
Figure 2-7 An illustration of the recorded highway in the highD [6].	39
Figure 2-8 Illustration of lane segments along a reference path (red) in OpenDrive. The reference path has been divided into three sections, each having a different number of lane segments in the traversal direction. It can be observed that a lane segment (-2 in Section 1 and Section 2) merges into its neighbour (-1 in all three sections) and eventually vanishes, leading to a re-adjustment of IDs in Section 3 [24].	43
Figure 2-9 The road layout is constructed by several lanelets and a single lanelet is highlighted. The roles of the bounds specify the driving direction, as the black arrow shows.	45
Figure 2-10 (a) is a Liblanelet map. Lanelet B is the right neighbour of lanelet A because the left bound of lanelet B is identical to that of lanelet A. Lanelet D and lanelet E overlap partly. (b) is the resulting directed graph showing the traversability of traffic agents in the Liblanelet map.	45
Figure 2-11 Schematic representation of a merging lane using OpenDrive and lanelets.	46
Figure 2-12 A Lanelet2 map example for one direction of a motorway	47
Figure 3-1 An overview of classification in this research.	51
Figure 3-2 Illustration of message passing in GCN and GAT separately. The same structure is used in GCN and GAT, a host node aggregating information from its three neighbour nodes without self-connection. In GAT, the aggregated node features from neighbours are assigned different weights. The higher the weights are, the more important the neighbour nodes are. In this illustration, α is the weight and $\alpha_1 > \alpha_3 > \alpha_2$	57
Figure 3-3 Illustration of vehicle trajectory prediction methods based on FF....	58

Figure 3-4 Illustration of trajectory prediction pipeline based on historical point cloud input and CNN methods. Point clouds at each time step are arranged in the 3D voxel grid, and historical voxel grids are concatenated in temporal sequence to form a 4D tensor. To collapse the temporal dimension, height information in the voxel grid is treated as channels in the 2D tensor, and 1D convolution is deployed to process the spatial data across the temporal dimension, equivalent to producing a single point cloud from all frames by weighting their contribution [51].	59
Figure 3-5 Illustration of LSTM prediction model with an encoder-decoder structure. The input to the encoder is a sequence of historical vehicle state data and the output is a fixed-length vector summarizing of the whole input. This one-hot vector is used as the initial state of the LSTM in the decoder. Along with the current state of the vehicle, the decoder runs recursively to create the output sequence step by step.....	61
Figure 3-6 Illustration of vehicle trajectory prediction model based on TF.....	63
Figure 3-7 Illustration of GAN and CVAE-based trajectory prediction in (a) and (b) separately. (a) the GAN-based method randomly samples the latent variable with a known distribution, e.g., Gaussian distribution. (b) the CVAE-based method creates latent variable space through training. In the testing stage, latent variables are sampled from this learned distribution of the latent variable space.	64
Figure 3-8 Illustration of the identification methods of SAs. The TA is in purple; the identified SAs are in dark blue and the irrelevant traffic agents are in light blue. In (a), eight SAs are identified based on whether they are directly adjacent to the target vehicle. The SAs are marked according to their relative positions to the target vehicle including behind (b), forward(f), left(l) and right(r). Some positions might not be covered by any SA, as expressed by the dashed contours. In (b), SAs are identified based on their Euclidean distances relative to the TA. In (c), a space grid is created and vehicles inside the grid cells are identified as SAs.....	67
Figure 3-9 Road layout maps could be represented in rasterized images or map graphs, as shown in (b) and (d) separately. The rasterized map in (b) is created from (a), where the drivable areas (in grey), lane markings (lane markings in light yellow, pedestrian crossings in white) and lane centrelines are filled with identical colours. The map graph in (d) is created from (c), showing the traversability of lane segments.	76
Figure 4-1 The mapping between bounding boxes and vehicle shapes is poor in some cases. Some bounding boxes do not overlap with the vehicles and contain the road surface (in red) [13].	105
Figure 4-2 Examples of intersections recorded in the INTERACTION [17]. ...	106
Figure 4-3 Four target scenarios in the inD [20].	108
Figure 4-4 A summary of traffic agent data concerning traversal times.....	111

Figure 4-5 The potential entrance and exit for the road users. And the potential manoeuvres are marked.....	112
Figure 4-6 The clustering results at Intersection A.	113
Figure 4-7 The clustering results at Intersection B.	114
Figure 4-8 The clustering results at Intersection C.	114
Figure 4-9 Used tools and related tasks.....	116
Figure 4-10 Typical definition of lanelets is listed. In (a), only the right boundary of the lanelet is assigned, leading to an incomplete definition of a lanelet. In (b), both left and right boundaries are given.	118
Figure 4-11 Disordered line points are shown on the right side, which forms a zigzag-like boundary. The order of these three points is then refined, which is shown on the left side.	119
Figure 4-12 The directions of Lanelet A and C are aligned with that of the lane while Lanelet B is oriented in the opposite direction. The lane will be broken in connection because no shared points can be found between the left/right boundaries of Lanelet B and those of either Lanelet A or Lanelet C.....	119
Figure 4-13 Pipeline of map data processing (a) and the customised data structure to store the map information (b).....	122
Figure 4-14 An illustration of sample extraction through sliding windows.....	125
Figure 4-15 An example of the 13by3 grid. The filled polygons stand that a SA has been identified in grid cells.....	126
Figure 4-16 The track point (red dot) will be matched to both Lanelet A and B because it is contained within both of them. This overlapping of map elements usually happens at merging or splitting lanes and at intersections.	127
Figure 4-17 The illustration of the state transition in the CV model at time $t-1$ and t	133
Figure 4-18 The illustration of the state transition in the CVTR model at time $t-1$ and t	134
Figure 4-19 Schematic of LSTM. C_t is the cell state of the LSTM cell. x_t is the input at the current time frame. h_t is the hidden state. σ is the <i>sigmoid</i> activation function.....	142
Figure 4-20 Schematic of GRU. xt is the input at the current time frame. h_t is the hidden state. σ is the <i>sigmoid</i> activation function.	143
Figure 5-1 Classification of road centreline calculation methods.....	152
Figure 5-2 Illustration of a combination of two-lane pieces. E_i are extreme points of the line segments. C_i are circle centers of the maximum circles. The red	

- straight dotted line denotes the ligature. And the curved dashed red lines are estimated centerlines in the ligature. 154
- Figure 5-3 Illustration of maximal inscribed circle types. Two types of constraints: the circle being tangent to a segment or passing by a point. In (a), the circle is constrained by 3 segments. In (b), the circle is constrained by 2 segments and 1 extreme point. In (c), the circle is constrained by 1 segment and 2 extreme points. In (d), the circle is constrained by 3 extreme points. In (e) the circle is constrained by 1 segment and 2 extreme points, with one of the two points lying on the segment. In (f), the circle is constrained by 2 segments and 1 extreme point, which lies on one of the two segments..... 156
- Figure 5-4 An illustration of a range search and a nearest-neighbor search, based on point R_3 and point R_4 separately. Blue line segments belong to the left lane border while red ones belong to the right lane border. In the range search, R_3 should be matched with L_3 , L_4 and L_5 , but not L_8 . In the nearest neighbor search, R_4 should be matched with L_5 , and L_6 , but not L_7 158
- Figure 5-5 Example of MDs that need to be filtered out. The circle C_1 is crossed by border segment E_6E_7 , thus is beyond the lane region. The contact point of the circle C_1 with segment E_1E_2 is on the segment extension, which has produced a circle beyond the scope of segment constraint. 159
- Figure 5-6 The left lane border, in blue, and the right lane border, in red, are oriented according to the direction of the arrows. The associated MDs are shown together with their contact points. 160
- Figure 5-7 An illustration of the centreline (solid red segments) estimated by the VT. The grey lines are Voronoi edges perpendicular to border segments. The solid blue segments are the resulting edges from inside-lane Delaunay triangulation, while dashed blue segments are edges of Delaunay triangles outside the lane polygon. N_1 and N_3 are the start and end nodes of the centreline separately, and N_2 is a branch node. The branch node has three connected edges, one of which is a residual spur (the red dashed line). 162
- Figure 5-8 Visualization of the results of the intermediate steps of centreline extraction based on DT. The lane border segments are marked in red. (a) is the distance map and the colours represent distinct distance values from the pixel to its nearest border segments. (b) is the ridge map and colours represent the ridge intensity. 164
- Figure 5-9 The first scenario in the inD. Sparse points are marked in blue. Lanelets are represented by polygons. Vehicle driving lanes and non-driving lanes (like pedestrian paths) are marked in light yellow and grey separately. Examples of a complete pedestrian path and a driving lane are bounded by solid and dashed red lines separately. 165
- Figure 5-10 Illustration of centreline extraction results of one lane. (a)-(d) are results in the straight part and are obtained by MD, DT (0.4m), VT (original points) and VT (1m) separately. (e)-(h) results in the normal bending part and

are obtained by MD, DT (0.4m), VT (original points) and VT (1m) separately.	168
Figure 5-11 Illustration of centreline extraction performance in the sharp corner. (a) is the target lane and its centreline benchmark. (b)-(d) are results from MD, DT (0.4m) and VT (1m) separately.....	170
Figure 5-12 The deviation from the calculated centreline to the benchmark centreline. The red, black and blue lines are from MD, VT and DT separately.	171
Figure 6-1 The structure of the proposed prediction model with the encoder- decoder structure. In this illustration, LSTM is used as an example in both trajectory encoder and decoder.	175
Figure 6-2 DEs in the going through case	179
Figure 6-3 DEs in the turning-left case	179
Figure 6-4 DEs in the turning-right case.....	180
Figure 6-5 Visualization of the prediction results in the turning (a) and going through (b) cases. Blue lines are ground truth. Red lines are historical trajectories. Yellow lines are predicted trajectories.....	182
Figure 6-6 The prediction results of an agent (Dataset ID 1, Actor ID 148) are illustrated here. The agent moves on the road around 20s (the x-axis on the left side). In the prediction error, prediction errors across the prediction horizon (4s) are plotted as the heatmap at each time step. To better show the agent dynamics, trajectory curvature (rad), speed (m/s) and heading angles (rad) are also plotted. On the right side, the trajectory of the agent (in red) and the trajectory curvature (in pink) are plotted.	183
Figure 6-7 Visualization of the prediction results in the turning (a) and going through (b) cases. Blue lines are ground truth. Red lines are historical trajectories. Yellow lines are predicted trajectories.....	184
Figure 6-8 The prediction results of an agent (Dataset ID 1, Actor ID 148) are illustrated here. The agent moves on the road around 20s (the x-axis on the left side). In the prediction error, prediction errors across the prediction horizon (4s) are plotted as the heatmap at each time step. To better show the agent dynamics, trajectory curvature (rad), speed (m/s) and heading angles (rad) are also plotted. On the right side, the trajectory of the agent (in red) and the trajectory curvature (in pink) are plotted.	184
Figure 7-1 The model (LSTM_CSP1) contains a shared trajectory encoder for TA and SAs, a Convolutional social pooling layer and a trajectory decoder. The input of the LSTM is the concatenation of the agent trajectory encoding and social context encoding. The hidden states of the LSTM will be initialised by zeros in the decoder.	189

Figure 7-2 In this structure (LSTM_CSP2), the input of the LSTM is the concatenation of social context encoding and TA's current status. The hidden states of the LSTM will be initialised by the trajectory encoding of the TA.	190
Figure 7-3 Demonstration for the module of CSP. The trajectory encodings of SAs are filled into the grid to form the social tensor. Then two convolutional layers and one max-pooling layer are used to extract the social context encoding.....	191
Figure 7-4 DEs across the prediction horizon for LSTM_CSP1 and LSTM_CSP2.	195
Figure 8-1 The grey background is the intersection surface and white lines are the lane markings. The red line is the historical trajectory of the TA. The yellow line and purple dotted line are the ground truth and predicted trajectory in the prediction horizon. 12 map elements are selected according the distance from the TA to the start point of the centreline and they are marked with pink polygons. For conciseness, centrelines are not visualised. Map element feature encoders are applied to process the centre points of map elements. The resulting map element feature encodings are imported into the agent-map attention module, along with the encoding of the historical trajectory, to produce the agent-map attention encoding. This agent-map encoding will be concatenated with the current status of the TA and serve as inputs of the trajectory decoder to produce the final prediction.....	203
Figure 8-2 Overall performance across the prediction horizon	209
Figure 8-3 Visualization of the attention results. The polygons with darker colours indicate a higher attention score, where more importance is attached compared to the others. The red lines are the historical trajectory of the TA, and the blue and yellow trajectories are the ground truth and predicted trajectories separately.....	210

LIST OF TABLES

Table 2-1 Datasets collected by on-board sensors	36
Table 2-2 Datasets collected by drone sensors.....	41
Table 3-1 A summary of backbone architectures	66
Table 3-2 A summary of interaction minning architectures.....	74
Table 3-3 A summary of road scene mining architectures	84
Table 3-4 A summary of fusion architectures	87
Table 3-5 Comparison of state-of-the-art model performance based on NGSIM	95
Table 3-7 Comparison of DL-based model performance on Argoverse	96
Table 4-1 Time-dependent variables for tracks in the inD	107
Table 4-2 Features of the four scenarios [20].....	109
Table 4-3 A summary of track numbers and components in four datasets.....	109
Table 4-4 The track number of different actor classes in the target scenario .	110
Table 4-5 The summary of the number of tracks of different manoeuvres	113
Table 4-6 An example of track data arranged in row.....	123
Table 4-7 A summary of observation and prediction horizons.....	124
Table 4-8 The statistics of actor samples in each scenario	127
Table 5-1 The Comparison of Results of 3 Methods	167
Table 6-1 The influence of the size of parameters	177
Table 6-2 The prediction performance with different LSTM inputs	178
Table 6-3 The results of using LSTM and GRU.....	181
Table 6-4 The FDE and ADE values by actor classes and manoeuvre classes	181
Table 7-1 The FDE and ADE values by actor classes and manoeuvre classes	193
Table 8-1 Results of model performance	207
Table 8-2 Comparative Results on inD measured by ADE/FDE.....	208

LIST OF EQUATIONS

(3-1)..... 54

(3-2)..... 54

(3-3)..... 88

(3-4)..... 88

(3-5)..... 89

(3-6)..... 89

(3-7)..... 89

(4-1)..... 128

(4-2)..... 133

(4-3)..... 133

(4-4)..... 133

(4-5)..... 134

(4-6)..... 134

(4-7)..... 134

(4-8)..... 134

(4-9)..... 134

(4-10)..... 135

(4-11)..... 135

(4-12)..... 135

(4-13)..... 135

(4-14)..... 135

(4-15)..... 135

(4-16)..... 136

(4-17)..... 136

(4-18)..... 136

(4-19)..... 136

(4-20)..... 136

(4-21)..... 137

(4-22).....	137
(4-23).....	137
(4-24).....	137
(4-25).....	138
(4-26).....	138
(4-27).....	138
(4-28).....	138
(4-29).....	138
(4-30).....	138
(4-31).....	139
(4-32).....	139
(4-33).....	139
(4-34).....	139
(4-35).....	139
(4-36).....	140
(4-37).....	140
(4-38).....	141
(4-39).....	141
(4-40).....	141
(4-41).....	141
(4-42).....	143
(4-43).....	143
(4-44).....	143
(4-45).....	143
(4-46).....	144
(4-47).....	144
(4-48).....	145
(8-1).....	204
(8-2).....	204

(8-3).....	204
(8-4).....	204
(8-5).....	204
(8-6).....	204
(8-7).....	208

LIST OF ABBREVIATIONS

Ac	Actor
ADE	Average Displacement Error
AM	Attention Mechanism
AV	Autonomous Vehicle
BEV	Bird-Eye-View
BP	Back-Propagation
CNN	Convolutional Neural Network
Conv	Convolution
CSP	Convolutional Social Pooling
CTRA	Constant Turn Rate and Acceleration
CV	Constant Velocity
CVAE	Conditional Variational Auto Encoder
CVTR	Constant Velocity and Turn Rate
DA	Drivable Area
DL	Deep Learning
DLT	Discrete Laplace Transform
Ds	Dataset
DSD	Driving Scenario Designer
DT	Distance Transform
ET	Execution Time
EV	Ego Vehicle
exitD	Exits and Entries Drone Dataset
FaF	Fast and Furious
FC	Fully Connected
FDE	Final Displacement Error
FF	FeedForward Neural Network
FPV	First-Person View
Fr	Frame
GAN	Generative Adversarial Network
GAT	Graph Attention Network
GCN	Graph Convolution Network
GF	Generative Framework

GPS	Global Positioning System
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
highD	Highway Drone Dataset
HPC	High Performance Computing
ID	IDentification
Idx	Index
IMU	Inertial Measurement Unit
inD	Intersection Drone Dataset
INTERACTION	INTERnational, Adversarial and Cooperative moTION Dataset
LS	Lane Segment
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MATF	Multi-Agent Tensor Fusion
MD	Maximal Disk
MLP	Multilayer Perceptron
MR	Miss Rate
ND	Naturalistic Driving
NGSIM	Next Generation SIMulation
NN	neural network
NV	Non-effective Vehicle
OGM	Occupancy Grid Map
OSM	OpenStreetMap
RL	Road Layout
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
round	Roundabout Drone Dataset
SA	Surrounding Agent
SAM	Steepest Ascent Method
Sce	Scenario
SV	Surrounding Vehicle
TA	Target Agent
TF	Transformer

TA	Target Vehicle
UTM	Universal Transverse Mercator
VRU	Vulnerable Road User
VT	Voronoi Tessellation
WOMD	Waymo Open Motion Dataset
XML	Extensible Markup Language

1 Introduction

1.1 Background

The development of autonomous driving technologies marks a significant milestone in the evolution of transportation. The core of this technology is to enable autonomous vehicles (AVs) to navigate safely and effectively, thereby improving overall driving safety. To achieve this goal, the AVs should have the ability to have an advanced understanding of the surrounding traffic scene during the tasks of risk estimation and trajectory planning.

In the early stage, risk estimation and trajectory planning in AVs have been predominantly focusing on the current state of traffic scenarios. Such an approach, while foundational, offers a rather shallow comprehension of the traffic environment, neglecting the dynamic and evolving nature of traffic conditions. In contrast, experienced human drivers make decisions by integrating both the present traffic scene and its potential evolution. They anticipate the future movements of other traffic agents, an ability that autonomous systems must emulate for enhanced safety and efficiency.

The advantages of enabling AVs to infer and predict the evolution of traffic scenes are manifold. Beyond the estimation of the immediate intentions of drivers, such as turning or going straight, it is essential for AVs to predict the trajectories of surrounding traffic agents. This capability is critical in identifying and alleviating potential conflicts, especially in complex driving scenarios.

Research in the domain of trajectory prediction for AVs has made significant strides, particularly in highway contexts. Due to the monotonous road layout, the research focus has been on limited manoeuvres like lane changing and cutting in. However, the transition from highway to urban environments introduces a new spectrum of challenges. Urban scenarios feature more complex interactions among vehicles and vulnerable road users. Besides, the road layouts are more diverse, hence more manoeuvre patterns could be identified than those in the highway scenarios, such as stopping, turning, and yielding. These complexities call for new approaches to trajectory prediction in urban scenarios.

A gap in existing research is the limited exploration of trajectory prediction in urban contexts, primarily due to the scarcity and complexity of these types of naturalistic driving data. Recognizing this shortfall, the motivation of this project is to do research in trajectory prediction for AVs in urban scenarios, leveraging naturalistic driving datasets. This study aims to study the influences made by neighbouring traffic agents and use additional road layout information to perform trajectory prediction.

1.1.1 Project Aim and Objectives

This project aims to develop a data-driven trajectory prediction model in the urban scenario. The model leverages a naturalistic driving dataset in an urban intersection to train the deep-learning (DL) model. Neighbour influence and road layout information are explored separately to improve the accuracy of the trajectory prediction.

The objectives are listed as follows:

1. Investigating various types of deep-learning-based methods used in trajectory prediction and state-of-the-art technologies to encode traffic agent interaction and map information.
2. Investigating several naturalistic driving datasets and selecting suitable datasets for model training and testing.
3. Analysing behaviour characteristics of target traffic agents and pre-processing and splitting track and map data for model training.
4. Exploring suitable methods for representation of road layout and designing a new centreline calculation method to refine the original road layout data; formatting the road layout data for DL purposes.
5. Proposing a trajectory prediction model with the input of historical data of the target traffic agent and comparing the model performance of different architecture variables.
6. Validating a trajectory prediction model with encoded neighbour interaction against the literature and identifying the appropriate model structure to integrate the agent interaction encoding and the agent's status data.

7. Designing a trajectory prediction model with an improved encoding of road layout information and evaluating the performance against the backbone LSTM.

1.1.2 Thesis Outline

This thesis is a “paper format” style, which is one of the standard formats at Cranfield University. The planned publications have been written in Chapter 3, Chapter 6, and Chapter 8, including their introduction, methodology, results, and conclusion sections. I decided to write an article-based thesis format because the project’s achievements had been arranged in a to-be-submitted form. To present the context of the thesis more clearly, except for the article-formatted chapters, other chapters were written from the perspective of the whole project, including the introduction of the project (Chapter 1), review of naturalistic driving datasets (Chapter 2), review of deep-learning-based trajectory prediction methods (Chapter 3), methodologies (Chapter 4) and conclusions (Chapter 9). Specifically, this thesis is organised as follows:

Chapter 1 - this chapter presents the project background, aim and objectives, novelties, significance and thesis outline.

Chapter 2 - this chapter investigates the accessible datasets that could be used to train the deep-learning-based trajectory prediction models. The analysis is carried out based on data recording technologies and the advantages and disadvantages of each method are discussed. Also, different map formats that are commonly used in traffic scene descriptions are introduced in this chapter.

Chapter 3 - this article-based chapter reviews the deep-learning-based vehicle trajectory prediction methods. The review is carried out in terms of trajectory prediction backbone, agent interaction mining and road scene mining. The review guides the selection of our trajectory prediction backbone and neural networks used to extract expected features regarding interaction and road scene mining.

Chapter 4 – this chapter summarises this project's methodology. It begins by introducing the dataset selection and pre-processing steps to prepare the data for model training. After comparison, the inD is selected for further analysis. The dataset is then processed to extract data samples, including track data and data used to describe the traffic scenario, like the neighbour and road layout information. Additionally, two prediction methods based on constant models are introduced, followed by an overview of deep learning-based trajectory prediction models, including their inputs, outputs, potential methods, and evaluation metrics.

Chapter 5 – this article-based chapter presents a novel lane centreline extraction method. The attached map files in the inD do not provide lane centreline information, which is important in our project to represent the road layout information. The map elements in the provided maps are also defined in a sparse way, which poses challenges to existing centreline calculation methods. To this end, a novel centreline calculation method based on maximal disks is proposed, which is proven to be accurate, time-efficient and storage-saving.

Chapter 6 – a backbone of trajectory prediction in this project is proposed in this article-based chapter. An RNN-based encoder-decoder structure is selected considering its good performance in handling sequential prediction tasks. Two RNN units, LSTM and GRU, are compared to study their performance and variable hidden state sizes are also tested. Besides, different inputs of the trajectory prediction model are compared. These comparisons help design the trajectory prediction backbone in this project.

Chapter 7 – the influence of agent interaction mining is discussed in this article-based chapter. A model replicated from the literature is tested. Further, a modification is made to the replicated model to further improve prediction performance. The improved model gives insight into how to concatenate heterogeneous context encodings and how to initialize the hidden states of RNN units in the decoder.

Chapter 8 - this article-based chapter describes the trajectory prediction model that takes advantage of road layout context to improve the trajectory prediction accuracy. Sparsely described map elements are used to describe the road layout

and road scene context is extracted by 1D convolution and agent-map attention. The results show that additional road layout information can help improve prediction accuracy.

Chapter 9 - this chapter presents an overall conclusion of this PhD project and looks ahead to possible future work.

The relationships between the objectives and the chapters of this project are illustrated in Figure 1-1.

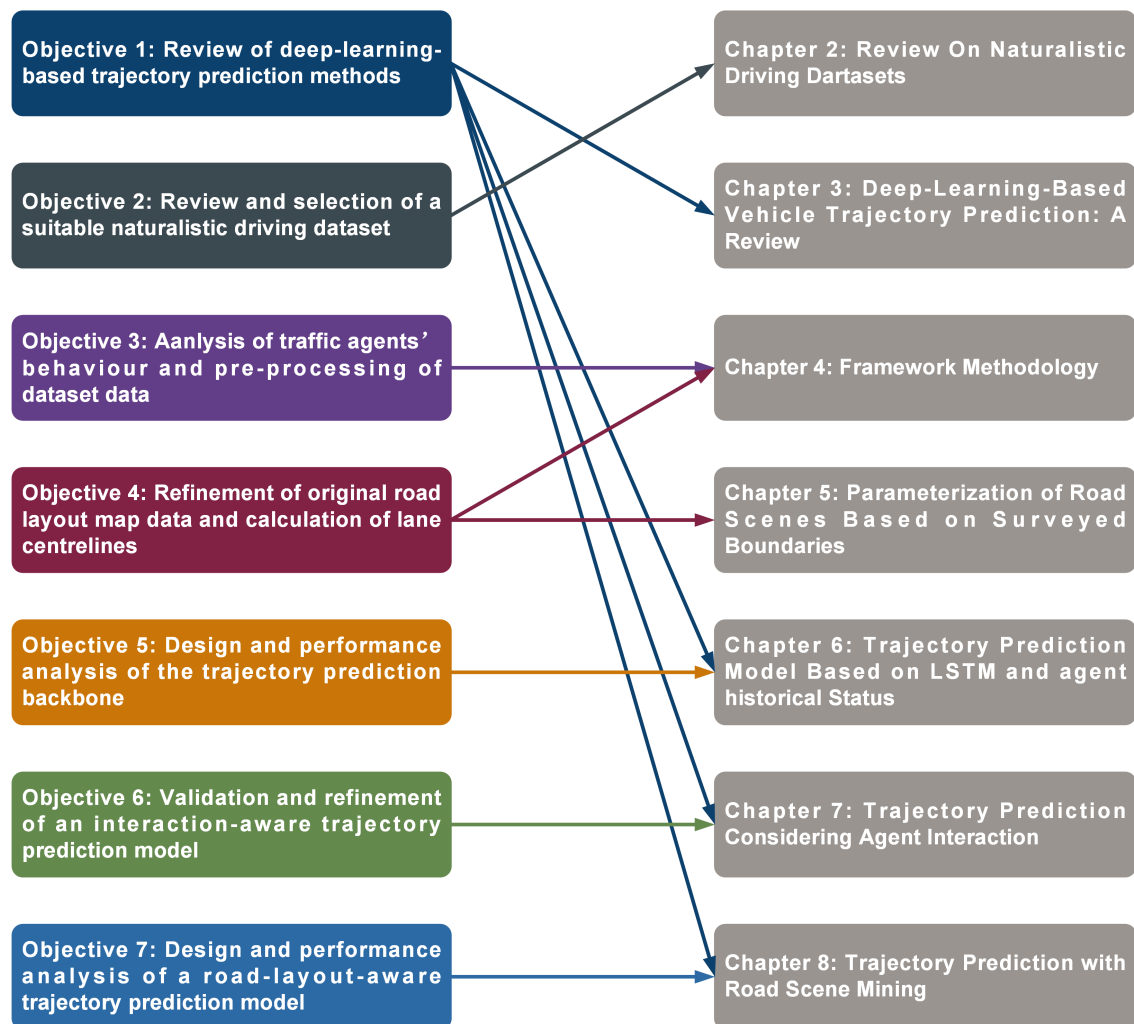


Figure 1-1 A diagram of relationships between the objectives and the chapters

1.1.3 Novelty of This Project

The novelties of this project are summarised as follows:

1. The literature review on DL-based trajectory prediction models explores the scope of the prediction components and their interconnections in the same prediction framework, including prediction backbone, vehicle interaction and road scene mining. A contribution of this work is its focus on the relationship between the traffic scene representations and the choice of suitable NNs, which has not been explicitly covered in prior work. This discussion is crucial for understanding complex architectural decisions and provides guidance for the design of improved solutions.
2. A novel approach for the extraction of the road centreline is proposed, based on extracting inscribed Maximal Disks (MDs) through lane tunnels. Instead of resampling the space, which is necessary for other methods, MDs are formed directly based on the constraining elements, either segments or their extreme points taken from the boundaries. It is time-efficient and storage-saving, especially when sparse maps are used. This is important because the sparse maps are a common representation of the road and environment with manual labelling, and the proposed method minimises the usage of memory and data storage.
3. The comparison of different trajectory prediction model inputs indicates the importance of the explicit calculation of relevant features, like the velocities of the traffic agents, to improve the prediction accuracy. A small architecture variation on the integration of the agent's historical encoding and agent interaction encoding has been made on one of the most promising models for encoding agent interactions [1] and this modification results in better prediction performances than the original. And the experiments on the integration of the agent interaction encoding and the agent's current status data guide the design of the model architecture when additional scenario data are also used as model inputs, such as the map encoding.
4. Instead of using fine-grained map information like existing work, sparse road-layout maps are used as additional inputs into the trajectory

prediction model for performance enhancement. Agent-map attention is used to efficiently and effectively extract useful map encodings to enable a map-aware trajectory prediction. The study on map-enhanced trajectory prediction shows the benefits of using sparse road layout maps for prediction performance enhancement.

2 Review on Naturalistic Driving Datasets

2.1 Naturalistic Driving Dataset

In the research on trajectory planning and prediction for traffic participants [2] [3] [4] [5], data-driven approaches have been applied widely. These approaches rely on large-scale naturalistic driving datasets to provide the training and testing samples. So, it is crucial to have datasets that provide precise, sufficient, and comprehensive traffic data.

To serve as a uniform validation platform for evaluating various developed trajectory prediction methods, the approach to recording naturalistic driving behaviour must meet several key requirements [6]:

1. **Unbiased Behaviour:** The behaviour of all road users should be natural and unaffected by the presence of measurement equipment. Ideally, road users remain unaware of the data collection, ensuring their behaviour is uninfluenced.
2. **Accurate Movement Data:** The dataset must comprehensively capture the movements and positions of all road users with high accuracy.
3. **Comprehensive Environment Data:** The recorded data should include all relevant information about the environmental conditions during the measurement period.

To capture these naturalistic driving data, the common recording methods take advantage of on-board sensors, infrastructure-fixed sensors and aerial sensors.

2.1.1 On-Board Sensors

Recording the motion data of the test vehicle as well as surrounding traffic participants with a bunch of on-board sensors is a common solution in the data collection. Usually, a combination of lidar sensors and cameras are used for scene perception, and GPS and IMU are used to get the dynamics and localisation of the test vehicle. These sensors have overlapped perception fields and sensor fusion technologies are used to get comprehensive and accurate traffic information.

One example is the KITTI Vision Benchmark suite [7], which includes various sensors such as a 64-beam Velodyne lidar, stereo cameras, GPS, and an IMU, as shown in Figure 2-1. This dataset covers a diverse range of roads and traffic scenes, providing data from a first-person view (FPV). However, it should be noted that the KITTI dataset only contains raw sensor data and does not include tracking data, necessitating the implementation of a tracker.

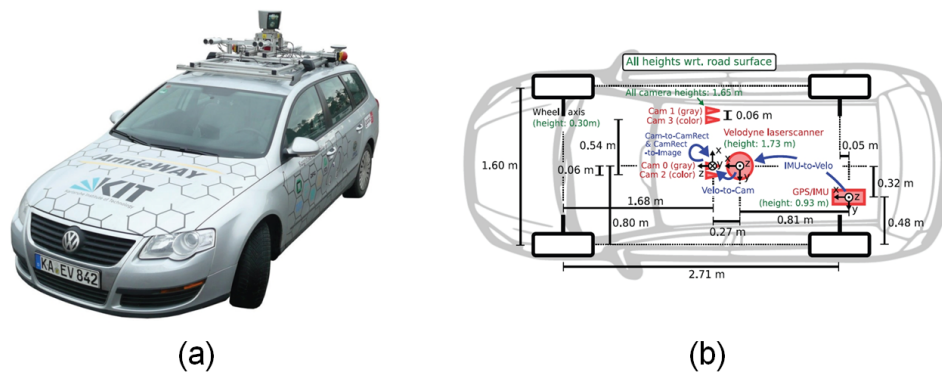


Figure 2-1 The test car used in KITTI data collection (a) and the layout of sensors (b) [7]

To better illustrate the surrounding scenario around the test vehicle, the collected data are often converted into a bird-eye view (BEV). One example is Argoverse. Totally two datasets have been released, named Argoverse 1 and Argoverse 2 separately [8] [9]. Argoverse 1 is created to support advancements in self-driving tasks like 3D tracking and motion forecasting. The data feature distinct urban driving challenges and local driving habits. In data collection, lidar sensors, ring cameras and front-facing stereo cameras are used, as shown in Figure 2-2.

Argoverse 1 includes a wide range of scenarios crucial for autonomous driving, such as navigating intersections, making unprotected turns, and changing lanes. It offers historical track information in 5-second segments, which is valuable for analysing these driving behaviours. Argoverse 2 is an upgrade of the former, containing more scenarios and extended track length of target vehicles (up to 11s).

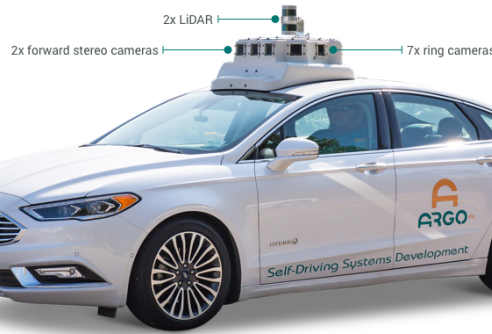


Figure 2-2 The test car used to record data in Argoverse 1 [8]

Another commonly used dataset is NuScenes. NuScenes [10] is the first dataset to utilize a comprehensive suite of sensors, including cameras, radars, and one lidar, providing a 360-degree perception around the vehicle. Collected in the dense urban environments of Singapore and Boston, the NuScenes prediction dataset contains over 1000 scenes, each lasting 20 seconds. It includes detailed information on road users, such as coordinates, size and orientation. Besides, map data of the traffic scene are provided.

The Waymo Open Motion Dataset (WOMD) [11] is primarily focused on urban scenarios with dense traffic. It features complex interactions and numerous critical situations such as merging and overtaking. WOMD includes multiple types of traffic participants simultaneously. The WOMD is collected by a combination of LiDAR sensors and high-resolution cameras, offering over 570 hours of total data.

These datasets gather data using a test vehicle that traverses various roads and traffic scenes, rather than remaining stationary. To analyse the behavior of nearby vehicles at intersections, one can isolate specific segments of these datasets. Through this approach, vehicle trajectories and their motion patterns at different intersections can be examined. However, each intersection must be labelled, and the recording vehicle drives through one single intersection very quickly. Consequently, studying individual intersections on a case-by-case basis is challenging due to the limited data available for each location [12].

To address these challenges, data can be collected using a well-equipped stationary vehicle positioned near intersections. An example of this approach is the ACFR dataset [12] which focuses on capturing naturalistic trajectories at unsignalised intersections. This system employs lidar scanners to detect road users within a range of up to 200 meters. Additionally, it has an on-board computer to perform real-time classification and tracking. The test car is shown in Figure 2-3.



(a)



(b)

Figure 2-3 The test car used in collecting ACFR [12]

A satellite view of one target intersection is shown in Figure 2-4. In the intersection, the two intersection approaches under study are labelled in blue, while the test car's location is marked in red. During recording, vehicles approaching the intersection from side streets were omitted due to sensor visibility limitations.

The method of recording naturalistic driving data in a vehicle review with on-board sensors mimics the perspective of a vehicle driving on the road and is more compatible with the actual driving situation.

However, data acquisition from a vehicle's perspective inevitably suffers from occlusion, potentially missing some target traffic participants. Additionally, the presence of a vehicle equipped with sensing devices may influence the behaviour of nearby human-driven vehicles, leading to less natural motion data being recorded.



Figure 2-4 The bird-eye-view of one recorded intersection, including the test vehicle (in red), and target intersection arms (blue arrows) [12].

The commonly used vehicle-perspective datasets are summarised in Table 2-1.

2.1.2 Infrastructure-Fixed Sensors

Naturalistic driving behaviour recording based on infrastructure-fixed sensors involves mounting sensors at dedicated roadside masts or street poles to continuously monitor a specific road segment.

An example of such a dataset is the Next Generation SIMulation (NGSIM) dataset, collected by the US Federal Highway Administration from 2005 to 2006 [13]. This dataset provides video data of the target traffic and has been widely used for research on traffic flow and driver models [6].

For the Interstate-80 section (San Francisco Freeway), seven cameras were mounted atop a multistory building at a height of 97 meters. For the US-101 section (Hollywood Freeway, Los Angeles), several cameras were mounted on an adjacent building at a height of 154 meters [14]. The cameras were tilted to cover the entire domain of interest, spanning a longitudinal length of 500 to 640 meters and a lateral width of five to six lanes [15]. The recordings have a total duration of 90 minutes. An illustration of the scenario from the camera's perspective is shown in Figure 2-5.

Table 2-1 Datasets collected by on-board sensors

Datasets	Sensors	Traffic Scenario	Motion Info	Map Info	Traffic Participants	Dataset Size
[8] Argoverse 1	lidar, ring cameras, stereo cameras	Dense urban	2D coordinate	OpenStreetMap, Rasterized Map of Drivable Area	Vehicle	over 1000 hours
[9] Argoverse 2	Camera, lidar	Dense urban	2D coordinate, velocity, heading angle	3D HD map	Vehicle, pedestrian, cyclist, bus, motorcycle	250,000 scenarios
[7] KITTI Vision Benchmark suite	lidar, stereo cameras	Urban	3D coordinate, orientation	-	Vehicle, pedestrian, cyclist	40,000 objects
[12] ACFR	lidar	Urban	Relative coordinate, Relative angle, velocity	-	Vehicle	Over 25,000 vehicles
[10] Nuscenes	Camera, radar, lidar	Dense urban	3D coordinate, heading angle	HD map	Vehicle, pedestrian, cyclist	over 333h
[11] Waymo Open Motion Dataset	Camera, lidar	Dense urban	3D coordinate, velocity, heading angle	3D HD map	Vehicle, pedestrian, cyclist	over 570h



Figure 2-5 An illustration of the infrastructure-fixed camera in NGSIM [14].

Roadside infrastructure sensors can provide an overview of traffic in the area of interest with high accuracy and most of the time the occlusions could be avoided. However, the sensors might be perceived by drivers and further cause confusion with traffic enforcement cameras leading to unnaturalistic driving behaviour [6].

Cameras could be installed in a high position to avoid this issue. But this might cause distortions of the observation for not in a perfect birds-eye view, which has been observed in the NGSIM. Also, to cover a considerable range of scenarios, several cameras are needed to work in a combined way, which poses further challenges in vehicle tracking and position alignment.

2.1.3 Aerial Sensors

A common approach is to use camera-equipped drones to record road users and measure their positions and movements from an aerial perspective. A quadcopter with high-resolution cameras hovers over the target area, recording traffic in high detail. The video footage is stabilised and rectified before being used to detect and track all traffic participants in the scene. The stabilisation is achieved using OpenCV by estimating transformations that align the background in each frame with the background in the first frame of the recording [6].

The Stanford drone dataset [16] was the first publicly available trajectory dataset recorded from a drone's perspective, specifically designed for analysing

pedestrian trajectories, as shown in Figure 2-6. It covers over eight locations on the campus and provides a recording of 9 hours. The dataset features a high percentage of pedestrians and cyclists, with cars accounting for only about 7%.

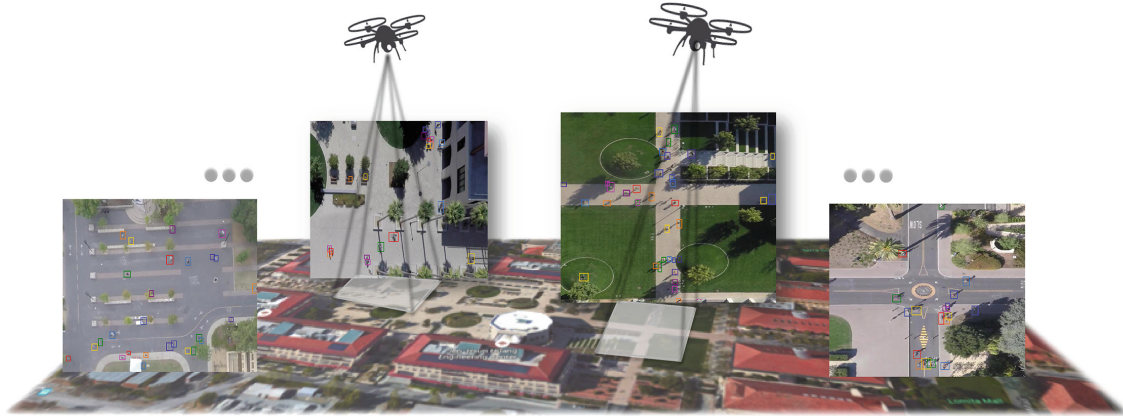


Figure 2-6 An illustration of scenarios that the Stanford drone dataset covered [17].

Following this project, several datasets were published covering broader driving scenarios. One such dataset is INTERACTION [17]. This dataset, gathered from multiple countries and regions with diverse driving habits, primarily encompasses a variety of scenarios such as ramps, roundabouts, and intersections. INTERACTION features a high level of vehicle interactivity within the scenes, challenging the trained models to effectively utilize interaction information. INTERACTION provides road users' track data and map data in lanelet2 format, which provides the road layout and corresponding semantic information.

The highD [6] is collected by a quadcopter equipped with a high-precision camera. The drone hovers directly above highways, capturing minimal vehicle sidewalls to reduce perspective distortions, as shown in Figure 2-7.

The highD dataset records over 110,000 vehicles across six different scenes. It has a total of 60 recordings and each recording averages 17 minutes in duration, with each vehicle appearing for an average of 13.6 seconds. The highD dataset features a wider range of speeds and higher maximum speeds compared to NGSIM, and it includes more comprehensive data on trucks.

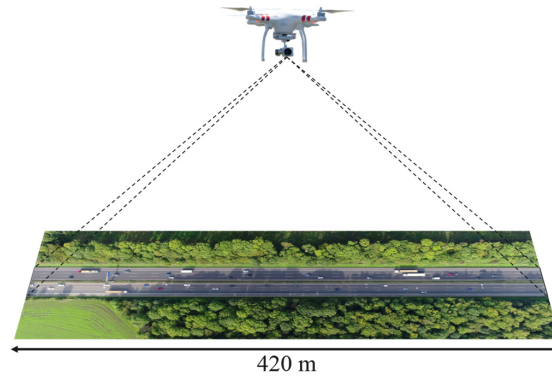


Figure 2-7 An illustration of the recorded highway in the highD [6].

The data collection and processing pipeline for the highD dataset involves the following steps [6]:

1. Do semantic segmentation using U-Net [18].
2. Create bounding boxes by detecting pixel clusters belonging to vehicles in the resulting segmentation map.
3. Annotate static objects such as lane markings, traffic signs and speed limits.
4. Connect detections in consecutive frames to tracks.
5. Match detections in two frames by distance or discard objects if no feasible match is found.
6. Use Rauch-Tung-Strieble Smoothing [19] and a constant acceleration model to smooth the trajectory of each vehicle.
7. Retrieve smooth positions, speeds and accelerations.

This technology is also applied in the following published drone datasets such as the intersection drone dataset (inD) [20], roundabout drone dataset (roundD) [21] and exits and entries drone dataset (exiD) [22].

The aerial perspective dataset is collected in a static region with no occlusion, allowing efficient capture of the motion states of all road users. Since the data acquisition device operates at altitudes ranging from 100 meters to several hundred meters, the drone is barely visible to passing vehicles. This results in completely uninfluenced, naturalistic driving behaviour [6] [20].

Although the drone applies flight stabilisation and gimbal-based camera stabilisation, there is still some drifting in the vehicle states, especially noticeable where the vehicles stop. Due to the requirement of stabilisation, the recording process is susceptible to interference from environmental factors.

The commonly used aerial perspective datasets are summarised in Table 2-2.

2.1.4 A Summary of Three Types of Recording Methods

2.1.4.1 Naturalistic Behaviours

Considering the naturalistic behaviours, the aerial sensors work best.

When recording data using onboard sensors, the behaviour of all road users might be influenced because test vehicles are often recognizable. The unusual appearance of externally mounted sensors can alert drivers, causing them to alter their behaviour due to the awareness of being part of a measurement. Consequently, the data collected from infrastructure-based sensors may not reflect natural behaviour.

In contrast, naturalistic behaviour is best preserved with aerial perspective data collection, as road users remain unaware of the measurement.

2.1.4.2 Coverage of Traffic Scenes and Conditions

Measurement data should capture the naturalistic behaviour of road users and include all relevant information to describe the identified scenarios with sufficient quality. A high-quality dynamic scenario description can be achieved from an aerial perspective, allowing for the accurate perception of all vehicles in every lane and both driving directions.

Aerial images captured by drones provide an unoccluded view of the recorded traffic scenario and ensure that algorithms can account for all road users. In contrast, on-board measurements can only capture the environment to a limited extent, with accuracy decreasing as the inspection distance increases, preventing a full scenario description.

Table 2-2 Datasets collected by drone sensors

Datasets	Sensors	Traffic Scenario	Motion Information	Map Information	Traffic Participants	Dataset Size
[6] highD	Drone camera	highway	2D coordinate	RGB aerial map image	Car, truck	Over 16.5h, 110000 vehicles
[20] inD	Drone camera	Unsignalized intersection	2D coordinate, velocity, heading angle, acceleration	Lanelet2	Car, truck, bus, pedestrian, cyclist	11500 trajectories
[21] rounD	Drone camera	roundabouts	2D coordinate, velocity, heading angle, acceleration	Lanelet2	Car, van, bus, trailer, truck, pedestrian, cyclist, motorcycle	13746 vehicles, over 3.6h
[22] exiD	Drone camera	Highway ramps	2D coordinate, velocity, heading angle, acceleration	OpenDrive	Car, van, truck	Over 16h, 69712 vehicles
[23]openDD		roundabouts		shapefiles	Cars, vans, trucks, buses, pedestrians, trailers, motorcycles, bicycles	Over 62.7h, 84774 trajectories
[17]INTERACTION		Urban intersections, highway		Lanelet2	Cars, pedestrians, bikes	Over 16.5h, 40054 trajectories
[16]Stanford Drone	Drone camera	campus			Pedestrians, bicycles, cars, skateboards, carts, buses	9h, 10240 trajectories

Infrastructure sensors can accurately measure the tracks of target objects on a specific road segment. However, objects passing close to the sensors can still cause occlusions, and additional scene distortion exists.

2.2 Map Formats Describing Traffic Scenarios

Static scenario information, such as road layout and traffic rules, is crucial in the development of path planning and prediction algorithms for autonomous driving for several reasons.

The geometry layout of the road and static markings like lane boundaries help autonomous vehicles navigate complex traffic environments safely and efficiently and improve the prediction accuracy of the behaviour of road users, as well as help autonomous vehicles perform in a more predictable way. Also, accurate maps with static information help the vehicle to localise itself within the environment.

Detailed road maps are crucial for autonomous driving. They facilitate the creation of a semantic environment model within the vehicle and help it understand both static and dynamic changes in its surroundings. Having prior knowledge of maps significantly enhances and speeds up the process of building a semantic map of the environment during vehicle operation.

Commonly used map formats include OpenStreetMap, OpenDrive, Liblanelet and Lanelet2.

2.2.1 OpenDrive

In OpenDrive, roads are defined along a reference path. Individual lane segments are created by offsetting a lateral distance from this reference path and assigning a customised length, as visualized in Figure 2-8.

A reference path is constructed by concatenating clothoids (Euler spirals), the curvature of which changes linearly with the path length [24], or just polynomials.

The reference path is divided into multiple sections, each containing a constant number of lane segments with uniform widths (see Figure 2-8). Lane segments

are defined by their type and width. The type tag differentiates driving lanes from other areas, such as sidewalks or parking spaces. Lane segments with negative lane numbers (IDs) have the same direction as the reference path, while positive IDs indicate the opposite direction. An illustration of lane definition is shown in Figure 2-8.

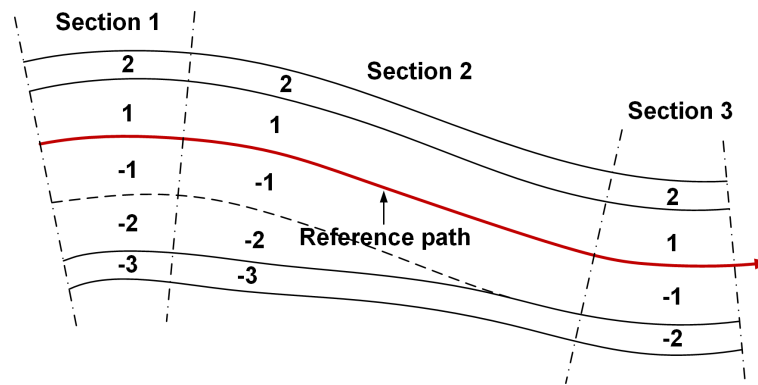


Figure 2-8 Illustration of lane segments along a reference path (red) in OpenDrive. The reference path has been divided into three sections, each having a different number of lane segments in the traversal direction. It can be observed that a lane segment (-2 in Section 1 and Section 2) merges into its neighbour (-1 in all three sections) and eventually vanishes, leading to a re-adjustment of IDs in Section 3 [24].

Partial reference paths can have none, one, or multiple successors, with the latter used to model junctions [24]. Connections to lanes within a road section are referred to as neighbours, indicating lanes that can be accessed through lane changes. These connections contribute to the overall road network.

The OpenDrive format uses Extensible Markup Language (XML) syntax and the data file ends up with the file extension .xodr. This format describes the geometry of roads, lanes, and objects. The road networks can be constructed based on either synthetic or real data [25].

2.2.2 OpenStreetMap

OpenStreetMap is a project initiated by University College London to enable users to have access to geographic data freely, developed based on the work of volunteers.

The core contents of OpenStreetMap are nodes, ways and areas. The nodes contain coordinate information, and all other geographical entities are created based on these primitives. Ways are defined by reference to a list of ordered nodes. Areas could be seen as a special form of ways, which are also a series of nodes, but the start and end nodes are identical. To contain semantic attributes of road features, key-value pairs could be created and linked to the above physical elements.

OpenStreetMap has good capabilities to represent topological information on roads. The path of a road is represented by a directed way, which is usually the centreline of the whole road, and the number of lanes attached to this directed way is given as the property of the way element. Further, the road network could be created by connecting the way elements. However, it has drawbacks in representing the geometry since a road is just represented as a single line and no lane information could be included, such as the lane width.

The map data are stored in OSM XML, a dedicated data transport format developed for the project [26].

2.2.3 Liblanelet

Lanelets are atomic, interconnected drivable lane segments [27], in which the traffic rules and connection relations do not change. Different from the lane segment definition in the OpenDrive, in which a lane segment is defined by offsets to its clothoid reference path, a lane segment (lanelet) is geometrically defined by its left and right bounds. These bounds are represented by an array of points in the form of polylines, as shown in Figure 2-9, and also specify the allowed driving direction. These bound points and bounds are called points and polylines in the Liblanelet. Since the bounds are polylines, arbitrary precise approximation of lane geometries is allowed.

With the definition of lanelets, a road network could be constructed by their connection. Two lanelets are longitudinally adjacent if the left and right end points of one lanelet are identical with the corresponding start points of the next lanelet in the driving direction [24]. And laterally adjacent lanelets could be identified if

the points of the left border of one lanelet are identical to the ones of the right border of another lanelet. These relationships can be seen in Figure 2-10. A direct road network graph could be established based on the connection of lanelets, which facilitates the path planning of autonomous vehicles.

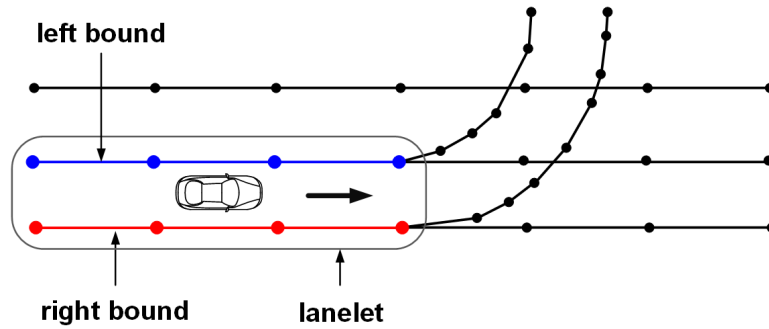


Figure 2-9 The road layout is constructed by several lanelets and a single lanelet is highlighted. The roles of the bounds specify the driving direction, as the black arrow shows.

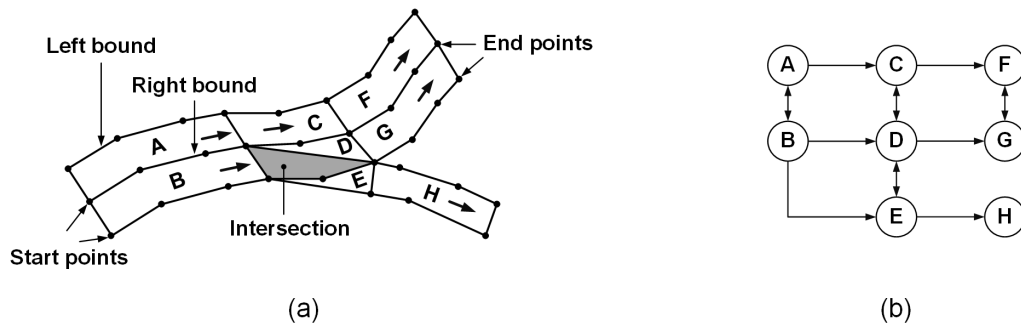


Figure 2-10 (a) is a Liblanelet map. Lanelet B is the right neighbour of lanelet A because the left bound of lanelet B is identical to that of lanelet A. Lanelet D and lanelet E overlap partly. (b) is the resulting directed graph showing the traversability of traffic agents in the Liblanelet map.

In the spatial dimension, two lanelets could overlap with each other. An example of overlapping can be seen in Figure 2-10, the grey polygon. This characteristic, along with lanelet adjacency, helps to better and more conveniently define lane merging and splitting. This is different from how merging and splitting are defined in OpenDrive. In OpenDrive, the merging of lanes has to be represented by gradually reducing their width to zero or split by gradually increasing the width from zero to avoid overlapping of lane segments.

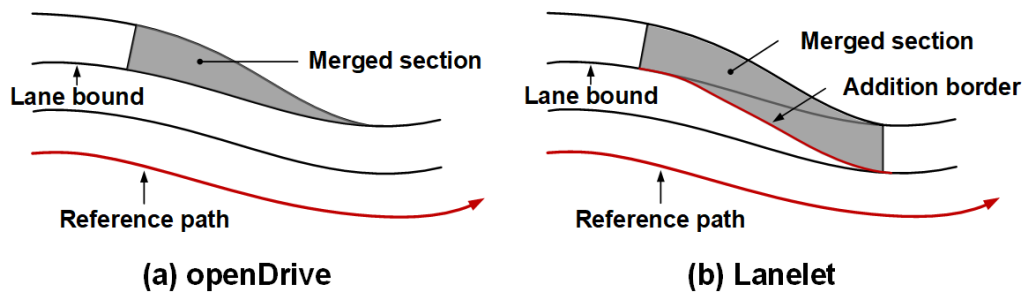


Figure 2-11 Schematic representation of a merging lane using OpenDrive and lanelets.

Besides geometry information, traffic regulations and road attributes could also be linked to lanelets, as separate regulatory elements. Thus, path planning could be performed based on lanelets with both geometry information and regulatory elements.

Considering the data storage format, lanelets are stored in OSM formalism – an XML-based file format.

The points, polylines and lanelets are related to the nodes, ways and relations in OSM. Also, regulatory elements, like traffic rules, are also stored as relations in OSM. Due to the utilization of OSM, all coordinates are given in the WGS84 coordinate frame.

2.2.4 Lanelet2

Lanelet2 is an extension and generalization of the lanelet and shares the same storage format as Liblanelet. Apart from basic geometry and semantic element lanelet, the concept of area is added. Areas are closed spaces in the map in which undirected or no movement is possible, like parking areas, squares, green spaces or buildings.

Lanelet2 is represented in a more structured way compared to the Liblanelet. The map is divided into a physical layer, a relational layer and a topological layer, as Figure 2-12 shows.

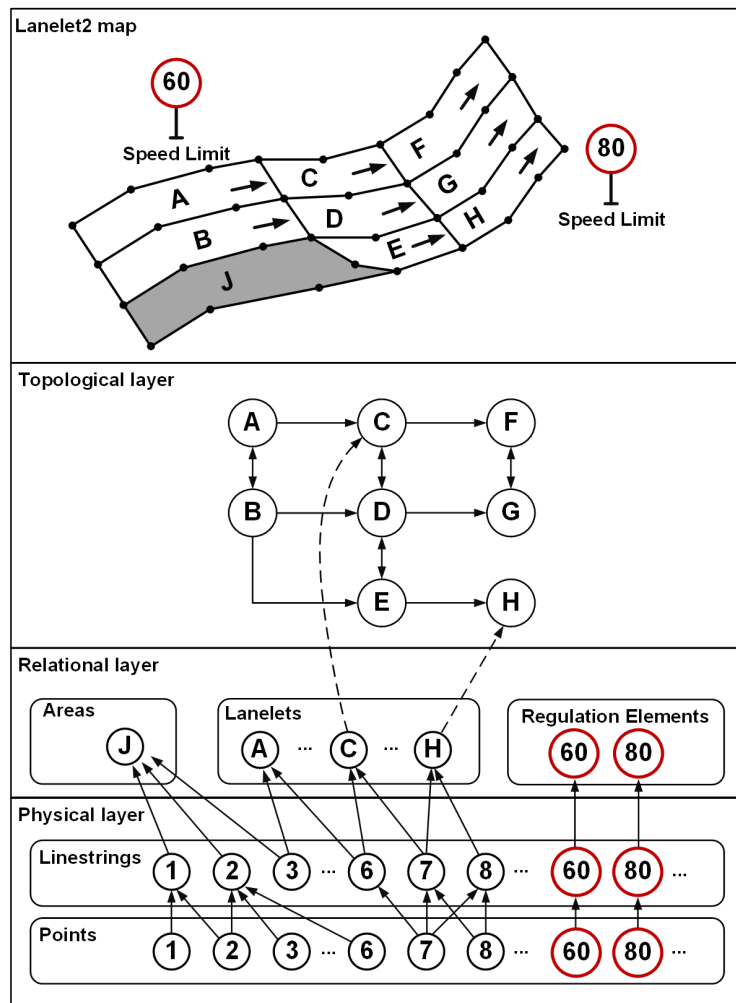


Figure 2-12 A Lanelet2 map example for one direction of a motorway

The physical layer includes real observable elements like points. They are the only primitives that have position information. And linestrings are directly constructed by an ordered array of points, representing road markings, curbs, fences, etc. in the real world or visual elements like lane boundaries inside the intersection. Compared to splines, they can be calculated efficiently and can be used to represent sharp corners. In the relational layer, there are lanelets and areas, as well as regulatory elements which include traffic rules. These regulatory elements could be assigned to physical elements and relational elements if the relevance exists. To represent the passable road network, the topological layer is established according to semantic and physical connections between lanelets and areas, as shown in Figure 2-12.

To improve the correctness of map physical data, geographic coordinates are transformed into the UTM (Universal Transverse Mercator) coordinate system, which is different from the WGS84 coordinates in Liblanelet.

Another difference from Liblanelet is that within a lanelet, the movement could be bi-directional, which is usually the case in the lanes for bicycles.

2.2.5 A Summary of Map Formats

OpenStreetMap, Liblanelet and its successor Lanelet2 are all free to users and are managed in a similar format. Liblanelet and Lanelet2 have more capabilities to describe lane-level information and thus could be used for autonomous driving. OpenStreetMap gives an overview of the road networks in a light format. However, as an open format, the quality of data in OpenStreetMap varies significantly since it is created and maintained by volunteers [26]. OpenDrive effectively represents road connections and lane-level information, but despite the open specification, there are no freely available libraries for interpreting and processing the data. So, OpenDrive is commonly used in industry while Lanelet2 in academia.

3 Deep-Learning-Based Vehicle Trajectory Prediction: A Review

3.1 Abstract

Vehicle trajectory prediction enables autonomous vehicles to better reason about fast-changing driving scenarios and thus perform well-informed decision-making tasks. Among different prediction approaches, deep learning-based (DL-based) methodologies stand out because of their capabilities to efficiently summarise historical data, infer nonlinear behavioural patterns from human driving data, and perform long-horizon prediction. This work reviews the DL-based methods that have shown promising results, organising them in terms of usage of the input data, separating the encoding of the target vehicle's historical data, surrounding vehicle's historical data, and road layout data. There are several ways to classify and analyse DL-based models. While prior works focus on types and architectures of neural network (NN) models, this chapter explores the relationships between the scope of the prediction components and the input data formats, as well as the connections with other components in the same prediction framework, including vehicle interaction and road scene mining. This information is crucial to understand complex architectural decisions and to provide guidance for the design of improved solutions. This work also compares the performance of the most successful prediction models, establishing that appropriate encodings of vehicle interactions and road scenes improve trajectory prediction accuracy, with the best performance achieved by attention mechanism and Transformer-based models. Finally, this work discusses future research directions, including considerations for real-time applications.

3.2 Introduction

Autonomous driving has the potential to improve road safety and traffic efficiency. For better navigation performance through complex driving scenarios, autonomous vehicles (AVs) require the capability of reasoning about fast-changing driving scenarios. One important part of this general problem is to predict the trajectories of surrounding vehicles on the road.

A simple approach for vehicle trajectory prediction is to rely solely on the current states of the target vehicles, based on the assumption that the vehicle's state transition is constant in the near future and the transition process, initialised by the vehicle's current states, is governed by the laws of physics. Models like Constant Velocity (CV) and Constant Velocity and Turn Rate (CVTR) are commonly used in this case [28]. Different types of estimators, like Bayes filters, are usually applied to deal with uncertainties in the observation of vehicle current states [29] [30]. These methods – based on the physics of expected and simplified vehicle dynamics – are computationally efficient but have poor long-term accuracy due to a lack of ability to reason about complex scene contexts like interactions between vehicles. Hence, they are often suitable in simple scenarios with short-term prediction horizons (less than 1 s) [31].

To provide additional contexts about the vehicle manoeuvres, many methods take into consideration their historical states. One option is to infer the most likely motion pattern of the target vehicle and select corresponding prototype trajectories [3]. The set of motion patterns includes possible manoeuvres at a given location. For a known road structure, they can be identified before the prediction tasks and matched to prototype trajectories, which are learned from previously observed trajectories or adhere to road geometries [32]. In the prediction stage, historical trajectories are matched with the predefined motion patterns, and associated prototype trajectories are selected to represent the future trajectories of the target vehicle [33]. One limitation with matching prototype trajectories is the difficulty of generalizing them for different road layouts because motion patterns and the related prototypes are trained for a specific road geometry. And this method cannot capture complex traffic dynamics like interactions between vehicles, leading to a loss of crucial context.

In the past few years, DL-based methods have gained attention for their ability to process complex traffic scene contexts and infer interactions among vehicles, and between vehicles and the road infrastructure. This contributes to effective long-term predictions. They are one of the most prominent methods in current literature. This chapter focuses on DL-based vehicle trajectory prediction

methods and seeks to provide a critical overview. In this work, the term backbone refers to the prediction architecture components to process the target vehicle's states and to combine the potential encodings of vehicle interaction and the road layout.

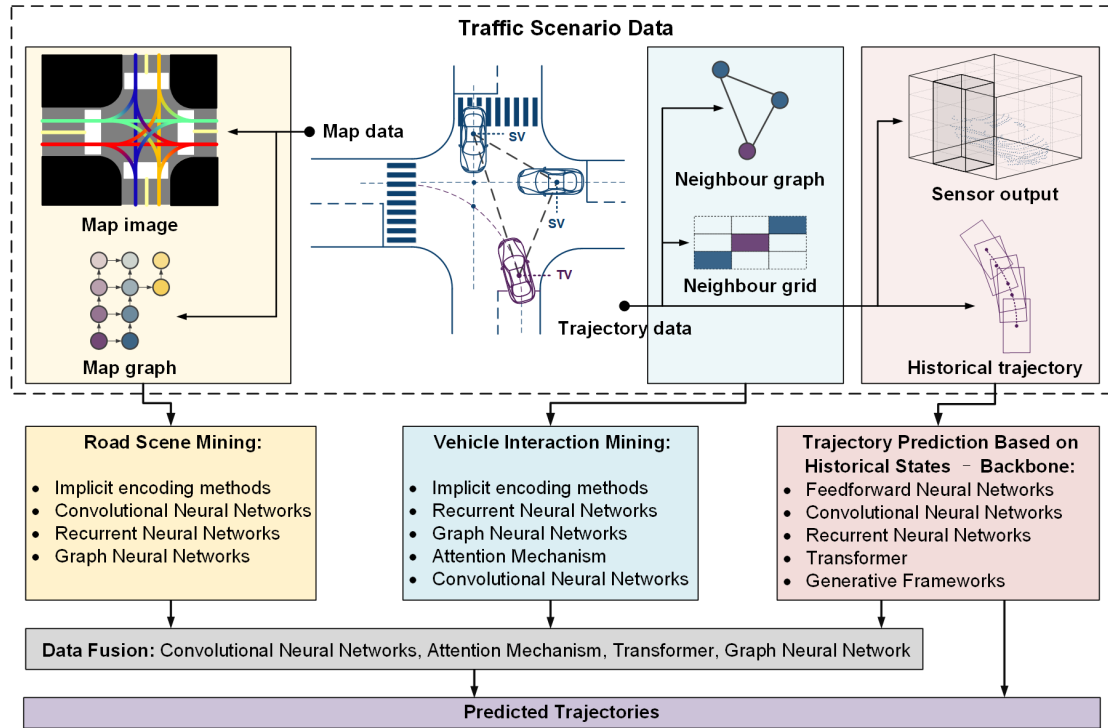


Figure 3-1 An overview of classification in this research.

This topic has previously been reviewed in several publications. Mozaffari et al. [34] have summarised deep learning-based methods and have discussed their superior capabilities in traffic scene understanding and trajectory prediction compared to conventional methods based on physics or classical machine learning [32], especially for a long prediction horizon. In their work, deep learning-based methods are classified according to input representation, output type and prediction methods. More specifically, in prediction methods, Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs) and their combinations are considered in detail. However, there is limited coverage of research works regarding the application of Graph Neural Networks (GNNs), despite the fact that, in recent years, they have been proven to be powerful in both vehicle interaction and map structure encodings. In [31], DL-based and

reinforcement learning-based (RL-based) methods are discussed, along with the physical-based and classical machine learning-based methods that have been mentioned in [32] [30]. In these works, significant attention has been paid to the classification of DL-based prediction methods in terms of Sequential Networks, GNNs and Generative Models. Details of the deployment of specific NNs are well summarised. A similar categorisation can also be found in [35]. Besides, optimisation settings of prevalent methods are listed, like the optimisers and learning rates during model training stages. These methods are also discussed by Gomes et al. [36]. In each classification, the discussion is carried out in terms of how vehicles' intentions and interactions are represented and used in trajectory prediction. In [37] main focuses are made on general trajectory prediction schemes such as diffusion models, generative models and sequential models, which generate an intermediate encoding by processing the original states and then use this encoding to produce the final predictions. Huang et al. [15] have reviewed the last five years' works on prediction and have summarised their improvements on the backbone prediction methods following the general representation of vehicles and map elements, integration of dynamic and static scene information and enhancement of prediction interpretability. The general framework of prediction methods is first introduced, including scene information, the scene context encoder, and the motion predictor. The commonly used input representations, like arranging scene information in grids or graphs, are also well illustrated in their work. However, there is no strong exploration of the relationship between the input representation format and the model chosen for input encoding.

It is a fact that the same NN could be used in multiple parts of one trajectory prediction model but with different performances and purposes. For example, CNNs could be used to extract temporal features of the vehicles. They could also be applied in the extraction of vehicle interactions and the extraction of map features, with minor modifications between these two applications. These applications of CNNs rely heavily on the format of input data and the type of features, temporal or spatial, that are expected to be extracted. In order to inform the reader of the performances of the best DL models and the latest research

trends, this work aims to analyse the relation between the architecture of the DL-based models for use in vehicle trajectory prediction, the prediction goal, the input information, and the corresponding data formats. The analysis is carried out in terms of trajectory prediction backbones, vehicle interaction mining and road scene mining. This area has not been explicitly covered in the literature and is one of the main focuses of this chapter.

This contributes to the body of literature by:

1. the review of popular DL-based vehicle trajectory prediction methods, focusing on the evolution of the methodologies according to the enhanced importance of vehicle historical states.
2. the analysis of the relations between the representation formats of driving scenario data and the chosen architecture in terms of trajectory prediction backbones, vehicle interaction mining and road scene mining, as shown in Figure 3-1.
3. the comparison of performance of the state-of-the-art DL-based trajectory prediction models, leading to some discussion on future research directions.

In the rest of this chapter, Section 3.3 introduces the basic concepts of DL-based vehicle trajectory prediction. Section 3.4 summarises the fundamental models of NNs used in the trajectory prediction tasks. Section 3.5 reviews DL methods used in trajectory prediction in terms of trajectory prediction backbones, vehicle interaction mining, road scene mining and vehicle-road fusion. The evaluation metrics and performance of state-of-the-art prediction models are compared in Section 3.6. Finally, the conclusion and the potential research gap are discussed in Section 3.7.

3.3 Problem Formulation of Vehicle Trajectory Prediction

The prediction of vehicle trajectories aims to forecast the future motion states of target vehicles over a fixed-length horizon based on the available information of the driving scenario, acquired by the autonomous ego vehicle. More specifically, the motion states are usually coordinates of the target vehicles at consecutive time steps.

In the driving scenario, the ego vehicle perceives the surrounding environment including other vehicles and road scenes. Among other vehicles, those vehicles whose trajectories are to be predicted are defined as target vehicles. Surrounding vehicles are defined as vehicles around the target vehicles which might have an influence on their moving motions. The remaining other vehicles are noted as non-effective vehicles which are assumed to have no influence on the target vehicles' behaviour.

Considering moving objects in the driving scenario are usually referred to as traffic agents, the ego vehicle, target vehicle, surrounding vehicle and non-effective vehicle are abbreviated as EA, TA, SA and NA for generalisation.

With a fixed historical horizon, the observation of vehicles (i.e. the road objects) is represented by O_{obs} , and other traffic scene information regarding road layouts and traffic rules (i.e. the road scene) are defined as S_{obs} . The vehicle trajectory prediction aims to obtain the motion states Y_{TVs} of the TAs over the prediction horizon based on O_{obs} and S_{obs} :

$$Y_{TVs} = \{Y_{t+1}^i, Y_{t+2}^i, \dots, Y_{t+p}^i\}_{i=1}^N \quad (3-1)$$

where Y_t^i is the predicted motion state of the TA i at time step t , N is the number of TAs and p is the length of the prediction horizon. More specifically, the Y_t^i is the 2D coordinates (x_t^i, y_t^i) . Considering the prediction uncertainty, the prediction task can be expressed in a conditional probability form: $P(Y_{TVs}|O_{obs}, S_{obs})$.

Due to the finite number of allowed vehicle manoeuvres at a given location, a multimodal prediction model can be used to inform the EA of the evolution and likelihood of different trajectories. A multimodal prediction produces several trajectory predictions at one time, indicating multiple possible sets of future vehicle motion states. In this case, the multimodal motion states of each TA could be represented by:

$$Y_{TV} = \{Y_{TV}^1, Y_{TV}^2, \dots, Y_{TV}^K\}, \quad (3-2)$$

where K denotes the total number of modalities.

3.4 Fundamental Deep Learning Algorithms for Trajectory Prediction

In order to realize the trajectory prediction, as well as vehicle interaction mining and road scene mining, the following deep learning methods are often used: Feedforward Neural Networks (FFs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Attention Mechanisms (AMs), Transformers (TFs), Generative Frameworks (GFs), and Graph Neural Networks (GNNs).

FFs are basic neural networks and are also known as Multilayer Perceptrons (MLP) or Back-Propagation (BP) neural networks [38]. They are generally used in vehicle trajectory prediction for the extraction of latent features in high-dimensional space and the conversion of high-dimensional data into outputs with physical significance. Their application as trajectory prediction backbone is summarised in Section 3.5.1.1.

However, the domain of sequential data, like historical trajectories, presents unique challenges. Here, RNNs specialise in processing these sequential data and capturing long-term dependencies that FFs are not designed to handle. In practice, gated RNNs, like Long Short-Term Memory (LSTM) [39] and Gated Recurrent Unit (GRU) [40], are commonly deployed in historical trajectory encoding and future trajectory generation, considering their better performance against gradient vanishing or gradient explosion in the backpropagation of training. RNNs' application in trajectory prediction is summarised in Section 3.5.1.3 and 3.5.2.2.

Compared to RNNs' ability to capture dependencies of sequential data, CNNs excel in extracting spatial features [41]. They are able to explore spatial interrelationships by processing image-like sensor inputs or bird-eye-view (BEV) representations of road scenes, which results in direct trajectory prediction or facilitates a more accurate prediction in cooperation with RNNs, by exploring spatial relationships between vehicles and road layouts and among traffic agents. The detailed application cases are summarised in Section 3.5.1.2, 3.5.2.5, 3.5.3.2 and 3.5.4.1.

Currently, the AM is popular in the application of trajectory prediction. It was originally proposed to improve the performance of the encoder-decoder translation model. The original encoder pre-processes the sequential inputs for the decoder as a fixed-length vector [42], which inevitably causes information loss due to its expressive limitation. The AM resolves this issue by enabling the decoder to visit each element of the input sequence and focus on particular parts [43]. The AM can be further classified into self-attention and cross-attention to learn self-presentation and allow knowledge integration separately. The TF is based on these two attention methods [44], within an encoder-decoder structure. Multi-head attention, a stacked of multiple self-attentions, is applied to extract a weighted sequence representation and cross-attention is used to transfer information from the encoder to the decoder. The application of AM and TF is shown in Section 3.5.1.4, 3.5.2.4 and 3.5.4.2.

In trajectory prediction, GFs are usually used to describe the underlying ambiguity of future behaviours of vehicles by introducing a stochastic latent variable while generating the prediction. The most commonly applied GFs in trajectory prediction are Generative Adversarial Network (GAN) [45] and Conditional Variational Auto Encoder (CVAE) [46]. GAN consists of a generator and a discriminator. The generator is to generate data with a combination of input and latent variables. These generated data will be mixed with the ground truth and then the discriminator estimates the probability that a sample of mixed data comes from the ground truth data rather than generated ones. The performance of the generator and discriminator could be iteratively improved and the probability of the discriminator making a mistake will be maximised at last [47]. Though a latent variable is also used in CVAE, its distribution is learnable. CVAE usually consists of an encoder and a decoder. In the encoder, a recognition network is created to produce the latent variable space based on the input during model training. In the decoder, latent variables are selected randomly from the produced latent variable space and are then combined with the input to do generation. Details in the application can be found in Section 3.5.1.5.

Other useful neural networks for trajectory prediction are the GNNs. GNNs are usually used while traffic elements are grouped by non-Euclidean metrics, like the interaction between vehicles and the connection of road layouts. In a graph, traffic elements serving as graph nodes and node edges are created by their (topological, semantic, legal, emotional, etc.) interconnections. GNNs update node features by message passing. Graph Convolution Network (GCN) and Graph Attention Network (GAT) are two significant branches in message passing. GCNs are inspired by traditional CNNs and the nodes update themselves by collecting features of their neighbours and performing aggregation like averaging and pooling [48], which is similar to the operation inside the CNN kernels. On the other hand, GATs assign different weights to the neighbours of a node by computing attention coefficients based on the features of each pair of nodes [44]. These coefficients are then used to update node features by a weighted aggregation, allowing the model to focus more on the most relevant information. An illustration of message passing in GCN and GAT is shown in Figure 3-2. Details in the application can be found in Section 3.5.2.3, 3.5.3.3 and 3.5.4.3.

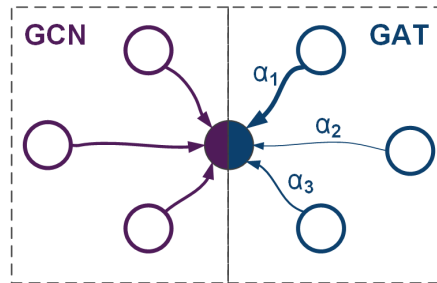


Figure 3-2 Illustration of message passing in GCN and GAT separately. The same structure is used in GCN and GAT, a host node aggregating information from its three neighbour nodes without self-connection. In GAT, the aggregated node features from neighbours are assigned different weights. The higher the weights are, the more important the neighbour nodes are. In this illustration, α is the weight and $\alpha_1 > \alpha_3 > \alpha_2$.

3.5 Classification of DL-Based Trajectory Prediction Methods

In this section, the trajectory prediction backbone is first reviewed in Section 3.5.1 to show a general framework of NN methods. Then NN methods for vehicle interaction mining and road scene mining are discussed in Section 3.5.2 and

Section 3.5.3 separately. At last, methods fusing vehicle interaction and road scene information are introduced in Section 3.5.4.

3.5.1 Basic Neural Networks for Trajectory Prediction Backbone

3.5.1.1 Feedforward Neural Networks

FFs project the historical states of the vehicles into a high-dimension hidden space for feature extraction and then infer future trajectories based on these hidden states [49] [50] as is shown in Figure 3-3.

The historical observations of the target vehicles usually contain speeds, steering angles and positions in the historical horizon. In [50], these data are fed into an FF directly, consisting of one input layer, one hidden layer and one output layer. The output of the FF is as simple as a lateral position point at a specific horizon, for example, the lateral position of the target vehicle at a given time (e.g. 1 s) in the future. Every time step, the FF processes a new set of historical samples and thus the predicted trajectory is continuously updated while the prediction model is in operation.

Though the FF is simple in structure and efficient in execution, the prediction horizon is constrained to a short horizon (e.g., 1 second) for its limited performance in exploring time-series dependencies of vehicle historical states.

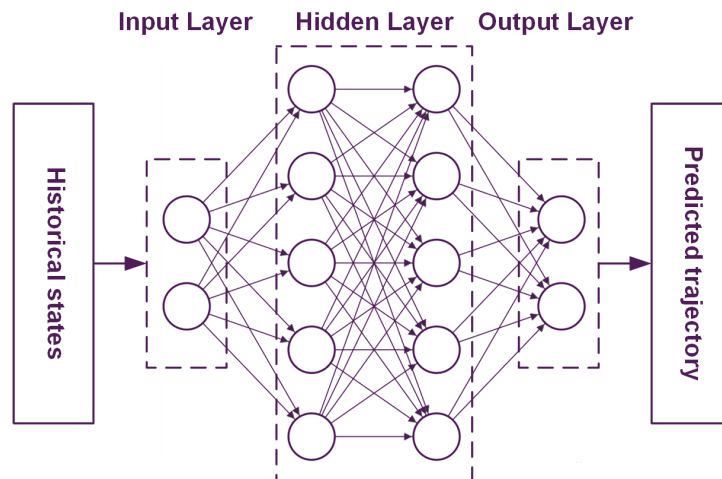


Figure 3-3 Illustration of vehicle trajectory prediction methods based on FF.

3.5.1.2 Convolutional Neural Networks

In trajectory prediction, CNNs are often associated with data inputs in the form of point clouds or RGB images. These data are common outputs of detection sensors and could be further processed to create a bird-eye-view (BEV) scenario describing both the static and dynamic scenes. The CNNs are then applied to extract spatio-temporal features and do the trajectory prediction in a short-term horizon [51] [52].

For convenience of CNNs processing temporal features from spatial data, the raw data from the sensors are usually concatenated with each other across historical time frames. FaF [51] uses a branch of convolution layers to predict the future tracklets, in the form of consecutive bounding boxes along the prediction horizon. 3D point cloud data are arranged in voxel representation (as shown in Figure 3-4) and grouped with an extended temporal dimension, ending up with a 4D tensor encoding an occupancy grid of the 3D space over several time frames. To process temporal features of the data, 1D convolution could be used to do data aggregation across the temporal dimension. After compression in the temporal dimension, the height dimension is treated as the channel dimension and 2D convolutions are applied across the X and Y dimensions to extract spatial features of the whole scenario.

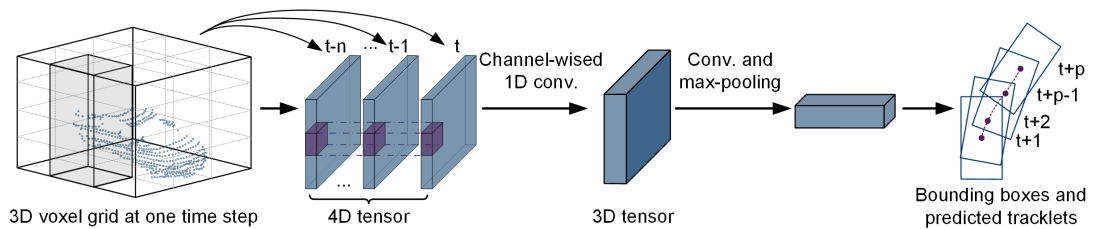


Figure 3-4 Illustration of trajectory prediction pipeline based on historical point cloud input and CNN methods. Point clouds at each time step are arranged in the 3D voxel grid, and historical voxel grids are concatenated in temporal sequence to form a 4D tensor. To collapse the temporal dimension, height information in the voxel grid is treated as channels in the 2D tensor, and 1D convolution is deployed to process the spatial data across the temporal dimension, equivalent to producing a single point cloud from all frames by weighting their contribution [51].

IntentNet [53] modified FaF's architecture by converting the voxel representation into an elevation map, and concatenating the time dimension to the height information, ending up with 2D grids. In this case, 3D convolutions are replaced by 2D convolutions to extract temporal and spatial features. A similar 2D convolution is applied to exploit semantic information from the map of the road layout. And outputs of these two convolution backbones are concatenated together and fed to the fusion subnetwork for intention estimation and final trajectory prediction.

Following this work, Casas et al. [54] extend the single branch network of PIXOR [55] to a two-stream network, such that the first stream processes LiDAR point clouds and the other processes road-layout maps, and their outputs are then concatenated along the channel dimension. This concatenation is further used to produce the bounding boxes of vehicles and to predict vehicle trajectories.

These research works unify object detection, short-term motion forecasting and tracking jobs together with an end-to-end fully convolutional approach, and motion forecasting is aimed to improve the performance of tracking. One limitation is that they are only performing well with a short prediction horizon, typically around 1s.

CNNs are valued in vehicle behaviour prediction for their capabilities of taking raw sensor data, like point clouds, as input and exploring their short-term temporal dependencies and spatial relationships. Though RGB images from cameras are also common inputs to AVs, they are usually processed through CNNs to perform object detection instead of trajectory prediction because pure CNNs are not efficient at extracting TA's temporal states and encoding them for trajectory prediction purposes. Thus, CNNs are seldom used as prediction backbones with RGB image inputs.

3.5.1.3 Recurrent Neural Networks

RNNs are often used to extract features that represent the motion of the TA across a historical horizon. The inputs are usually historical state sequences, including positions and velocities.

A single LSTM is used in [56] to predict the future position of the TA as a regression task. The purpose of the LSTM layer is to abstract a meaningful representation of the input time series, while its output is fed into dense layers to extract the final prediction: the future longitudinal velocity and lateral position. Additionally, the absolute states of the target vehicle are repeated and directly fed to the dense output layer, bypassing the LSTM to allow the model to focus on variations from the current states. In [57], in addition to the trajectory prediction LSTM module, another LSTM module is used to extract the lateral driving intentions of the TA. This intention encoding is then concatenated with the sequential longitudinal features of the TA and fed to the trajectory prediction LSTM module, to predict the lateral deviation and longitudinal acceleration across the prediction horizon. In [58], an encoder-decoder structure is used, as shown in Figure 3-5. The encoder works as an observation stage, extracting some features from historical data that are fed to the decoder to generate the prediction trajectory.

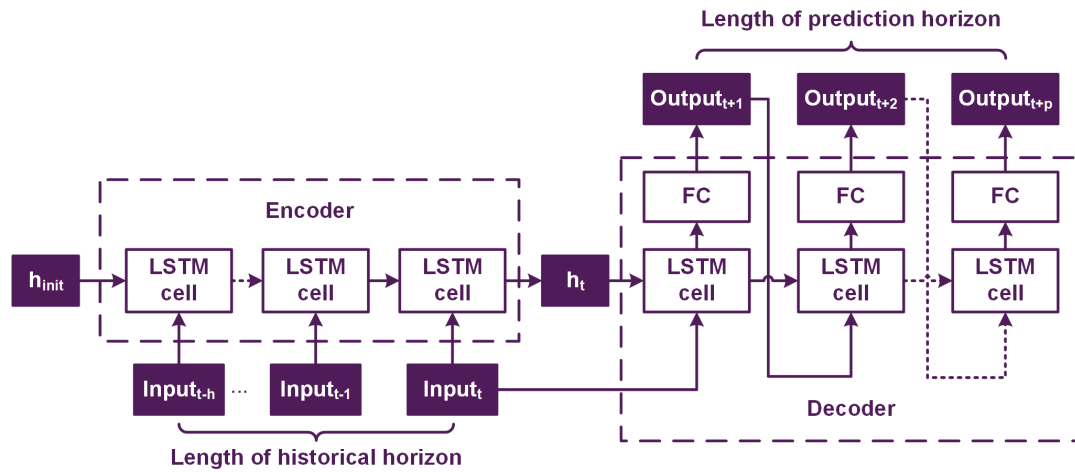


Figure 3-5 Illustration of LSTM prediction model with an encoder-decoder structure. The input to the encoder is a sequence of historical vehicle state data and the output is a fixed-length vector summarizing of the whole input. This one-hot vector is used as the initial state of the LSTM in the decoder. Along with the current state of the vehicle, the decoder runs recursively to create the output sequence step by step.

To have a better encoding of the historical data, LSTMs could be stacked together. Park et al. [59] use an encoder-decoder LSTM to predict the probability of occupancy on a grid map and apply a beam search algorithm to select k most probable future trajectory candidates. The LSTM encoder consists of two stacked LSTM layers and takes the latest trajectory samples from the as well as the state information of the EV to produce a fixed-length vector which captures the temporal structure of the past trajectory. Based on this fixed-length vector, the LSTM decoder, also a 2-layer stacked LSTM, generates the future trajectory on the occupancy grid map (OGM).

Similar to the method in [57], Deo and Trivedi [60] proposed a separate manoeuvre classification module based on a single LSTM to predict manoeuvre class probabilities based on the backbone of the encoder-decoder structure. These manoeuvre encodings are appended with the vehicle trajectory encodings from the trajectory encoder. Then the decoders, different for each manoeuvre, use this information to predict the final position of the TA.

Although RNNs are some of the main neural networks associated with data time series analysis and prediction, they have some deficiencies in modelling spatial relationships such as vehicles' spatial interaction and image-like data such as road scene contexts. This explains why sophisticated solutions using RNNs usually exploit additional methods to compensate for the weakness of a single RNN.

3.5.1.4 Transformer

Another neural network structure that is skilled at learning historical data is the Transformer (TF). One difference in processing sequential data between TF and LSTM is that TF needs additional positional encodings in the input because it is permutation-invariant and cannot distinguish the time flow [44].

The TF is made up of an encoder and a decoder, as shown in Figure 3-6. In the application of trajectory prediction, the encoder creates a summary of the complete time-series input, e.g. the historical states, by constructing two vectors of keys (K) and values (V) through self-attention. The decoder runs recursively to

predict future positions by focusing on appropriate information in the input as well as previously generated predictions [61]. Alternatively, in the decoder, the historical states of the target agent could serve as the input as well [62].

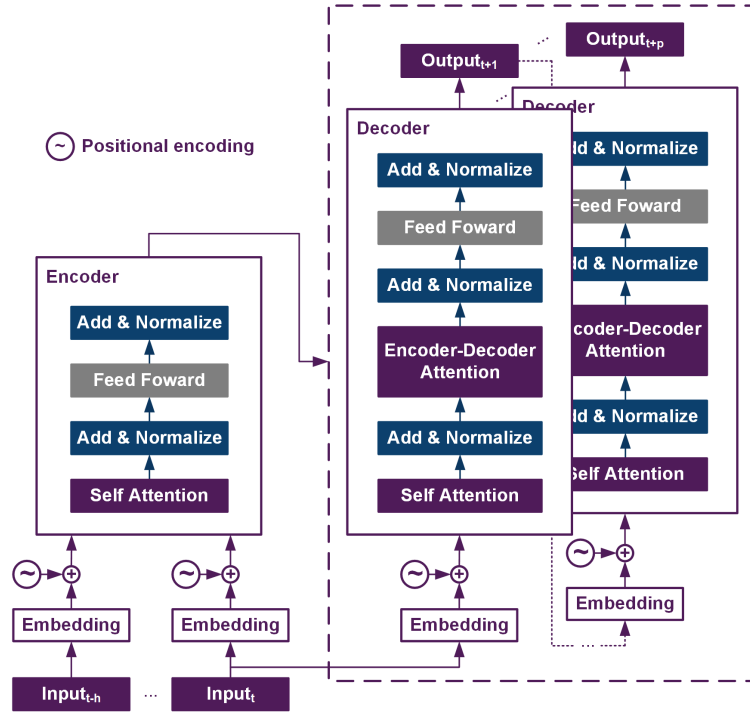


Figure 3-6 Illustration of vehicle trajectory prediction model based on TF.

In [63], a structure of dual TFs is used to predict future trajectories of the TA. The inputs of the first TF are the historical positions of the TA and the interaction information, consisting of relative states of the SAs, whereas its output is the manoeuvre intention. This intention information is concatenated with the input of the historical trajectory of the TA and fed to the encoder and decoder of the second TF to produce future predictions.

LSTM may undergo gradient vanishing or information loss while back-propagating the signal sequentially from the current to the earliest time step, but TF could learn from every single input of the historical data because it keeps the complete time-series encoding. Results have shown that the ability of TF has outperformed LSTM in pedestrian trajectory prediction [61].

3.5.1.5 Generative Frameworks

When treating trajectory prediction as a regression work, one of the limitations of pure RNNs is that they cannot consider the multi-modal character of human drivers. Future predictions can be inherently ambiguous and have uncertainties as multiple plausible scenarios can be explained under the same past situation (e.g., a vehicle heading toward an intersection with decreasing velocities can make turns or just perform giving way). Thus, learning a deterministic function that directly maps historical data to deterministic output will under-represent potential prediction space and easily overfit training data. Moreover, a naively trained network with a simple loss will produce predictions that average out all possible outcomes, e.g. the model tends to do a prediction which is an average performance of right-turning and going-through trajectories while the vehicle is driving in a right-turn and straight lane.

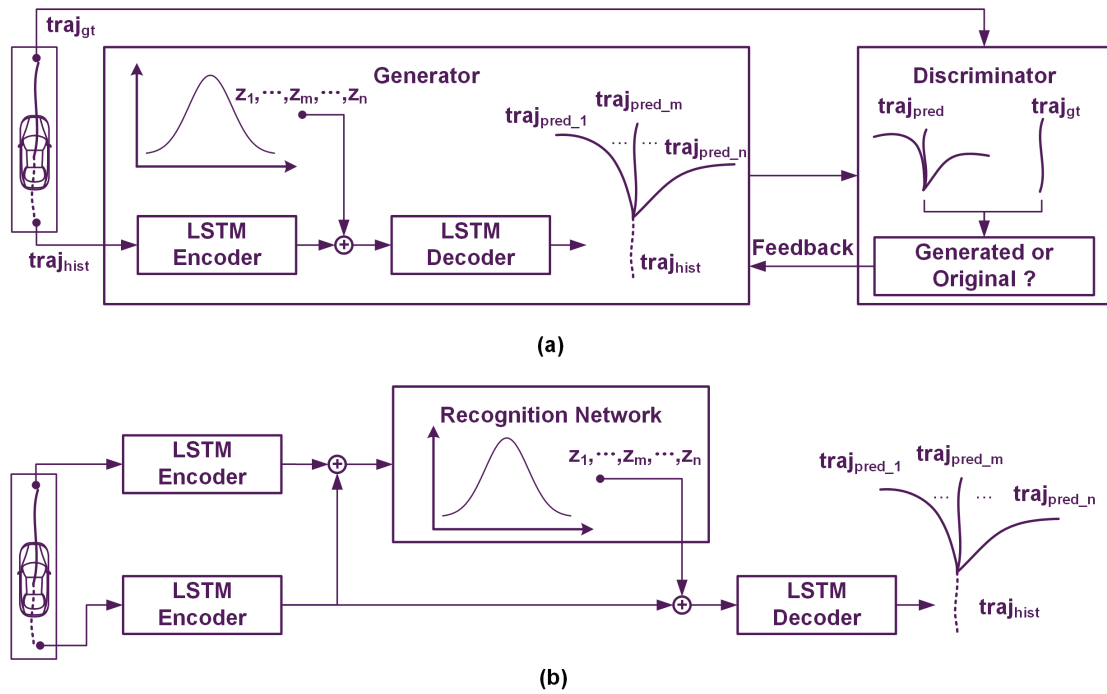


Figure 3-7 Illustration of GAN and CVAE-based trajectory prediction in (a) and (b) separately. (a) the GAN-based method randomly samples the latent variable with a known distribution, e.g., Gaussian distribution. (b) the CVAE-based method creates latent variable space through training. In the testing stage, latent variables are sampled from this learned distribution of the latent variable space.

GFs solve these issues by introducing stochastic latent variables in the prediction models. The most commonly applied GFs in trajectory prediction are Generative Adversarial Network (GAN) and Conditional Variational Auto Encoder (CVAE).

Lee et al. [64] have created a CVAE-based trajectory prediction method based on GRU. In the training phase, the CVAE encoder encodes the past and future trajectories of an agent through two separate GRUs. Then these two encodings are concatenated to produce the mean and standard deviation values in the latent variable space, with FCs. In the decoder, the latent variable produced by the encoder is reshaped to be combined with the encoding of the past trajectory through an element-wise multiplication and is fed into a GRU to generate the candidate prediction trajectory. In the inference phase, the encoding of the past trajectory is combined with randomly sampled latent variables to produce multiple candidate prediction trajectories. In [65], a similar method is proposed. The difference is that LSTM layers are used and, in the CAVE encoder, the trajectories of SAs are added to the trajectory of the TA to consider potential interactions.

In terms of the combination of CVAE and CNN, Dulian et al. [66] have used a CNN-based encoder to extract features of road scenario data, including separate layers of drivable areas, lanes and traffic agents. The latent variable space is then created based on these layer encodings, along with the agent state encoding. In [67], the encoder is LSTM-based and similar to that in [65], and a U-Net-like CNN [18] is used to process the road scene context, working as part of the CVAE decoder.

A summary of each backbone NN can be found in Table 3-1.

3.5.2 Neural Networks for Interaction Mining

Vehicles move on the road according to the actions of SAs, adjusting the heading angles and velocities to avoid potential crashes and negotiating with others to improve traffic efficiency. To this end, if the interaction between the TA and its SAs could be considered, the trajectory prediction accuracy could be potentially improved.

Table 3-1 A summary of backbone architectures

NNs	Inputs	Works	Pros	Cons
FF	historical state sequences	[50] [49]	<ul style="list-style-type: none"> • Simple in structure and efficient in execution • Useful to project mined traffic scene context into final trajectory prediction 	<ul style="list-style-type: none"> • Limited performance in exploring time-series dependencies of agents' historical states
CNN	Point clouds	[51] [53] [54] [55]	<ul style="list-style-type: none"> • Capable of keeping the spatial relationship of the traffic scene input data, enabling modelling of the interactions between agents and the road scene context 	<ul style="list-style-type: none"> • Only performing well with a short prediction horizon, typically around 1s, because of its limitation on extracting temporal features
RNN	historical state sequences	[56] [57] [58] [59] [60]	<ul style="list-style-type: none"> • Powerful in data time series analysis and prediction 	<ul style="list-style-type: none"> • Deficiencies in modelling spatial relationships such as agents' spatial interaction and road scene contexts • Undergoing gradient vanishing or information loss while back-propagating the signal sequentially from the current to the earliest time step
TF	historical state sequences	[61] [62] [63] [68]	<ul style="list-style-type: none"> • Skilled at learning historical data from every single input of the historical data • Easy to realize multi-modal prediction 	<ul style="list-style-type: none"> • Needing additional positional encodings in the input because it is permutation-invariant and cannot distinguish the time flow • Having more trainable parameters and achieving limited performance with a small training dataset
GF	historical state sequences	[69] [64] [65] [66] [67] [18]	<ul style="list-style-type: none"> • Introducing stochastic latent variables in the prediction models to ease the production of multi-modal predictions • The distribution of the latent variables can be predefined or learned 	<ul style="list-style-type: none"> • The selection of latent variables in the inference stage might be arbitrary, and not able to model the uncertainty in driving intentions, which might produce unrealistic trajectories • latent variables are assumed to have a specific distribution which might not aligned with the traffic scene

SAs are usually identified if they have a potential influence on the motion of the TA. Adjacent vehicles in the eight directions around the TA are usually identified as direct neighbours and considered SAs [60] [65]. Often, the vehicle in front of the forward vehicle is also identified as SA [56]. See the vehicle ff in Figure 3-8 (a).

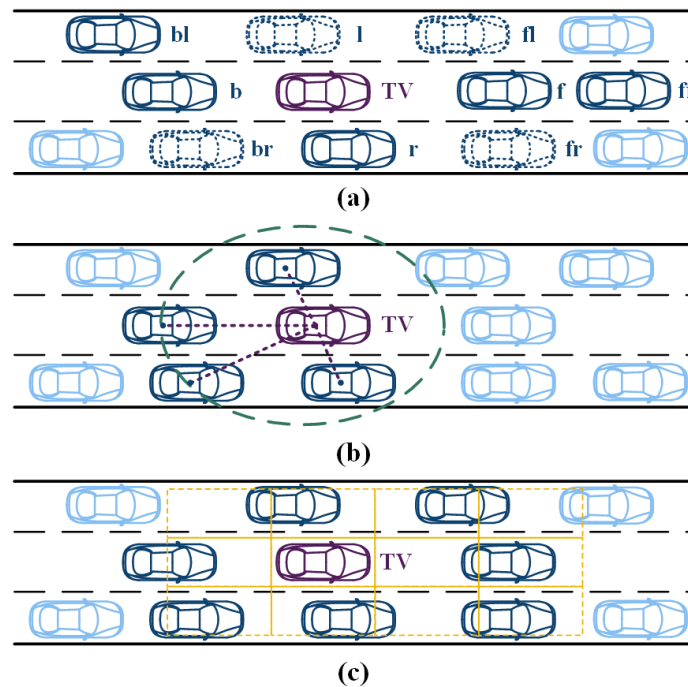


Figure 3-8 Illustration of the identification methods of SAs. The TA is in purple; the identified SAs are in dark blue and the irrelevant traffic agents are in light blue. In (a), eight SAs are identified based on whether they are directly adjacent to the target vehicle. The SAs are marked according to their relative positions to the target vehicle including behind (b), forward(f), left(l) and right(r). Some positions might not be covered by any SA, as expressed by the dashed contours. In (b), SAs are identified based on their Euclidean distances relative to the TA. In (c), a space grid is created and vehicles inside the grid cells are identified as SAs.

Taking into account the moving speeds, a rectangle field extending in the driving direction centred at the TA is considered a region of interest and all moving objects in this field are considered SAs [1] [70]. In this case, SAs are usually identified by setting a grid centred on the TA based on the Euclidean distance,

as shown in Figure 3-8 (c). The above definitions of SAs are usually used in simple scenarios, where the SAs move in the same direction as the TA, like highways.

However, there exists some certain limitations on this neighbour topology. First, it requires considerable storage to maintain information on identified neighbours across a large perception range, particularly in both forward-driving and traversal directions. In complex scenarios, such as large unsignalized intersections, SAs approaching from the sides of the TA can significantly influence its behavior. Capturing traffic from all intersection arms would require a grid of substantial size, leading to high storage costs. Second, due to the coarse-grained nature of the grid cells, multiple SAs may end up occupying the same cell. Although each cell is typically sized to fit a standard vehicle, SAs can still overlap within a single cell, requiring strategies for either merging them or distributing them across adjacent cells. Third, the rectangular grid shape does not conform to the actual road layout. This misalignment can lead to the inclusion of irrelevant SAs, particularly on curved roads where vehicles from opposite lanes may be included. In [71], the SAs are identified in a polar system. More generally, a star-like neighbour topology could be established as shown in Figure 3-8 (b).

3.5.2.1 Implicit Encoding

With identified SAs and their dynamic information, the interaction between SAs and the TA could be explored. One solution is to tail the SAs information just behind the TA's as a comprehensive input into the prediction model directly. This method fuses the SAs' interaction with the TA in an implicit way.

For the SAs, the most used information is the position. Both global coordinates [49] [60] and coordinates relative to the TA [56] [59] are used. The relative relations between vehicles are easier to be recognized by human drivers and this information provides more direct relationships than the absolute information. Meanwhile, the speeds of SAs are also used as input for the prediction model [56] [49] [59]. The speed information is not used alone but together with the position information, indicating the potential change of distance relationships. Other

information is also sometimes used in the inputs, such as the heading angle of the vehicles [58], yaw rate [59], acceleration [49] and time to collision [56].

Since the speed and acceleration are the first-order and second-order derivatives of the position, NNs could extract these features from the historical position data, but providing them directly allows for a reduction of the size of the model and the training requirements. There is some research using the control information of the TA like steering wheel angle [50], but this information is not easily measured for SAs.

In the implicit interaction fusion, no explicit process of computing the interaction feature exists, despite relative states to the TA. While this approach is easy to implement, it has certain limitations. NNs may struggle to effectively capture the interactions among the TA and SAs from the complex input arrays, which include the original states of both the TA and SAs. This challenge becomes more pronounced when the input arrays are high-dimensional, incorporating variables like speed, position and heading, especially when a large number of SAs are involved.

Thus, analysis of the interaction between vehicles as a separate procedure and producing a high-dimensional intermediate encoding can be concatenated with the TA states to do trajectory prediction. Such high-dimensional intermediate encoding is produced by analyzing the spatial relationship between the TA and its SAs by models like CNNs and GNNs. This is more explicit than simply appending the original states of SAs to the model input. Apart from the spatial relationship among SAs, the historical features are also extracted by pre-processing the historical states with models like CNNs and RNNs.

3.5.2.2 Recurrent Neural Networks

With identified SAs, the TA could be paired with each one of them to model the respective interaction. In [72], this pair-wise interaction is encoded by LSTM. Firstly, two different LSTM layers are used to separately encode the historical trajectories of the TA and its SAs. Then the historical encoding of the TA is paired with that of the SAs. Further, this concatenation is processed by a stacked LSTM

to further encode the impact of spatial interaction between the TA and its SAs. In [73], GRU encoders are used to model the pairwise interaction between the TA and each of the SAs. In each pair, two GRUs with shared parameters are used to encode the individual manoeuvre histories and dynamics. The two GRU encodings are then concatenated to represent the pair-wise interaction.

3.5.2.3 Graph Neural Networks

If the SAs are identified with a star-like structure, GNN could be used to propagate the information between the TA and its SAs. The TA and SAs in the star-like structure could form a graph, where each of them serves as graph nodes and their potential interactions form the graph edges. The connection could be bi-directional or unidirectional. An illustration of star-like vehicle interaction is shown in Figure 3-8 (b).

In [54], a fully connected directed graph is created based on potential interactions between the TA and its SAs, as well as the interactions among the SAs. The traffic agents are represented by bounding boxes and act as nodes in the graph. The node states are the convolution results of the fixed-size feature map. During message passing, concatenation of two connected nodes is first created and then processed by MLP to produce the message. Then the node updates its states by aggregating messages from all of its local neighbour nodes with an ordering-invariant, feature-wise max operator. By this feature-wise max operation, the model could figure out the interplay for each pair of vehicles in a bidirectional fashion. In [74], GAT is used to characterize the interdependences of heterogeneous traffic agents, including vehicles, bicycles and pedestrians. Traffic agents are treated as graph nodes, and within an identified TA and SA pair, the attention coefficient is calculated by matrix product based on the feature encodings of the nodes. Then the node feature is updated by the weighted sum of its neighbours. GAT is also used in [75] while the attention coefficient is calculated by normalized attention operation among all connected neighbour nodes [76]. In [77], convolutional layers are used to extract node features and GCN is used to aggregate node features from neighbours for the purpose of encoding interactions. In [56], the node features from the neighbours are weighted by the reciprocal of the

sum of the distance between the two nodes. This operation is designed to emphasize the influence of the SAs by their relative distance to the TA. The adjacency matrices of most GNN-based interaction mining methods are established through prior knowledge, like setting matrix values to be reciprocals of geographic distances. This static matrix lacks the adaptive learning capability to depict the evolution of agent interactions through time. In [78], a learnable adaptive graph is used to resolve this deficiency. More specifically, an adjacency matrix is initialised by the element-wise multiplication of the learnable embedding array of all agent nodes and their transpose. Then softmax is used to determine if a connection exists.

With these graph-like structures, the spatial relationships among the TAs and SAs can be explored. However, this structure only establishes spatial connections among vehicles at the current time frame. In [79] [80] to integrate the influence of the neighbours both in spatial and temporary dimensions, a spatio-temporal graph is created where edges across the historical time frames and through the spatial space are established. More specifically, at each time step, spatial edges are created among spatial-adjacent traffic agents; for the same traffic agent, temporal edges are produced to connect its states in adjacent frames. Based on these temporal and spatial edges, an instance layer is created for message passing and interaction capture among individual nodes. The interaction among traffic agents is modelled by a weighted sum of neighbour features, which is further concatenated with the temporal feature to update the node feature. Additionally, a category layer is established to identify traffic agents of the same type and a super node is used to summarise the shared behaviours of the same category by averaging node features in the whole layer. This feature is then transmitted back to the instance layer to form final interaction encoding, by connecting individual nodes to its corresponding super node. A simplified spatio-temporal graph can be established by creating a separate spatial graph at all temporal time steps [81] [82]. The spatial graph describes the interaction among vehicles and graphs across the time domain enable a dynamic description of interaction variation. To better evaluate the diverse influence of each SA across the time domain, an adaptive adjacency matrix is used in [81]. In the adjacency matrix, the edge weights between one node and its neighbours are calculated by normalized attention operation based

on node-neighbour encoding concatenations. This adaptive adjacency matrix has the ability to capture interactions among vehicles that are not explicitly covered by intuitive manual linkages [76].

3.5.2.4 Attention Mechanism

The influence of the SAs on the TA is not identical. For example, the SA following the TA might have less influence on the decision-making of the TA than its front SA or the SA that wants to merge into the lane of the TA. To identify the SAs that have more influence on the TA and better encode interactions, attention-based methods could be used. These methods often work in a kind of one-directional information transfer way, in which the interactive features of SAs are aggregated and linked to the TA.

In [83], the historical states of all agents in the scene are encoded with a sequential combination of 1D convolution and a variation of GRU. Then these encodings are used to create the Q, K and V vectors for the AM. The Q vector is generated based on the TA's encoding, while K and V vectors are based on the SAs' encodings. The normalized dot product of Q and K vectors creates an attention map and then this attention map is used to pool their value features into a context vector, which encodes the influence of the SAs on the TA. A similar structure is used in [84]. GRU is used to encode historical states of the vehicles in the scene and the final results of SAs are assembled into a global embedding matrix. Then attention weights among the TA and SAs are calculated by dot production of TA's hidden states and the generated global embedding matrix, which is further used to mine the interaction among vehicles. In [85], the historical states, future trajectories and reference path waypoints are encoded with the temporal convolutional layer and LSTM. Those encodings are fed into the linear layers for mapping Q, K and V vectors in the attention. A self-attention structure is then used to capture the interactions between agents in the past time domain. A similar strategy is also used in [86]. Considering the SAs are arranged in the occupancy grid, K and V vectors are mapped from the grid map instead of individual SAs in [87].

3.5.2.5 Convolutional Neural Networks

Another popular solution to deal with interactions from multiple neighbours is to use a social pooling operation [88]. In social pooling, a grid map is created, which is centred on the TA. Then each cell in the grid map is filled with states of SAs. The vehicle states could be their original dynamic information or high-dimensional states, processed by NNs like LSTMs. If multiple SAs occupy the same grid cell, the grid cell will be populated with aggregation of their states by an operation like adding. This social pooling preserves the spatial information of SAs and enables spatially proximal traffic agents to share information with the TA.

However, these grid cell states are concatenated end to end before being fed into the prediction model, which loses the spatial characteristics. To overcome this issue, CNN is used to capture the spatial inter-dependencies of the social pooling tensor. In [1] [89], convolutional and pooling layers are used to learn the locally useful features within the spatial grid of the social tensor. Two convolutional layers and one pooling layer are used to obtain the social context encoding. In [90], an additional MLP is used to process the output of the pooling layer to obtain the final interaction vector. Multi-Agent Tensor Fusion (MATF) encoding and decoding is introduced in [91]. In the encoding part, a social tensor, augmented with convolutional encoded scene context channels, is fed to a U-net-like [18] fully convolutional network to fuse interaction among agents and between agents and scene context while keeping spatial locality [34]. This combined method is also used in [70], but the AM is used to select the most noteworthy SAs at each time step. In [71], the SAs around the TA are identified in the polar coordinate. The relative position and velocity are concatenated with each vehicle's LSTM hidden state and processed independently by an MLP. Finally, the MLP output is max-pooled element-wisely to compute the pooling vector of the TA. In this way, the inter-dependencies of the motion of SAs are captured without being restricted to a specific grid size [92].

A summary of NNs for agent interaction mining is illustrated in Table 3-2.

Table 3-2 A summary of interaction minning architectures

NNs	Inputs	Works	Pros	Cons
implicit	Matrices to contain both TA's and SAs' states	[49] [56] [58] [59] [60]	<ul style="list-style-type: none"> • No extra NNs needed to process interactions • Easy to arrange data and simple in structure 	<ul style="list-style-type: none"> • Challenging to capture interactions because the input vector might be too large for the NNs to effectively extract and summarize the relevant features
RNN	historical state sequences	[72] [73]	<ul style="list-style-type: none"> • Simple to implement 	<ul style="list-style-type: none"> • Restricted to mine pair-wise interaction
GNN	Agent node feature matrix and adjacency matrix	[54] [74] [75] [76] [93] [78] [80] [79] [81] [82]	<ul style="list-style-type: none"> • Flexible to define connections among vehicles • Able to build connections across both the spatial and temporal dimensions • Flexible to choose message passing technologies, such as multi-step aggregation, bi-directional or unidirectional aggregation, attention-based aggregation 	<ul style="list-style-type: none"> • Storage inefficient because the number of SAs might vary and sometimes this ends up with a very sparse matrix
AM	Matrices of TA's and SAs' states	[83] [84] [85] [86]	<ul style="list-style-type: none"> • Able to mine the interaction according to the learned importance of SAs • Ordering-independent arrangement of SAs 	<ul style="list-style-type: none"> • Additional NNs needed to extract features before being taken as AM input
CNN	Grid maps with SA's states	[18] [70] [1] [71] [94] [88] [89] [90]	<ul style="list-style-type: none"> • Efficient in the operation of feature extraction • Able to preserve the spatial information of SAs and enables spatially proximal road actors to share information with the TA and capture the spatial inter-dependencies of the social pooling tensor 	<ul style="list-style-type: none"> • Requirements of representing SAs in a grid-like structure • Computational intensity in converting SAs' information to a fine-grained image

3.5.3 Neural Networks for Road Scene Mining

Vehicle motion is constrained by map-relevant information such as road structures and traffic rules. For example, the vehicles will follow the geometry of driving lanes and manoeuvre turning according to the lane connectivity and giving-way rules. More and more prediction models consider the fusion of map information to improve prediction accuracy.

3.5.3.1 Implicit Encoding

One way to explore the influence of road geometry on trajectory prediction is to represent the vehicle states relative to the road elements. In [95], the index of the occupied lane and the deviation from the lane centreline are considered part of the model input. In [57], the relative displacement to the lane markings and lane centre lines, as well as the first and second-order derivatives, are used as inputs of an LSTM model.

3.5.3.2 Convolutional Neural Networks

One popular method to process road scene contexts, including road geometries and traffic dynamics, is to form an image-like representation of traffic scenarios and deploy CNNs for spatial and temporal feature extraction. Basic ideas for creating image-like representations are to discretize the road scene around the TA into regular grids in Bird-eye-view (BEV) and further digitize the road scene with numerical values or RGB values to form multiple-channel grid maps or rasterization images [96] [83] [97] [98]. In [96] [99], the RGB images are used and they are created through rasterization, where different static contexts of the vector road-layout map are assigned distinct RGB colours.

In [96], a multiple-layer rasterization image is created to represent the traffic scenario around the TA. The traffic scenario is represented by pieces of vector layers, representing traffic elements like driving lane polygons, road boundary polylines and directional lane centreline polylines. The generated grids overlapping with the traffic elements are assigned distinct RGB values to make them prominent, as shown in Figure 3-9 (b). At last, a rasterization image could be obtained by stacking RGB images on top of each other in the sequence from layers that represent macro areas such as road polygons to layers that represent

finer structures like lane centrelines, where the subsequent images will overlay the content of the previous one. A similar technology is also used in [83]. Different from overwriting subsequent images, a multi-channel image is created to store the rasterized road scenes without loss of spatial features. For example, the completeness of the lane markers will not be affected when they are obstructed by agent boxes since these two features are stored in different channels.

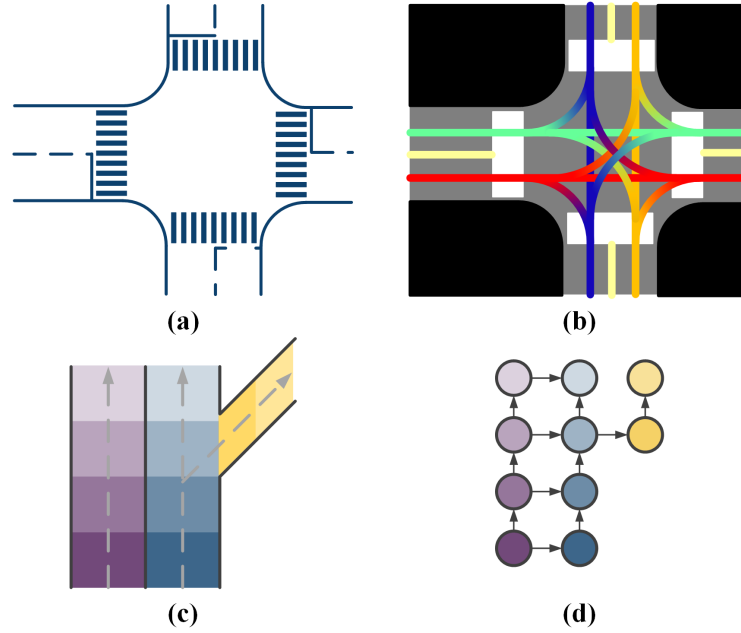


Figure 3-9 Road layout maps could be represented in rasterized images or map graphs, as shown in (b) and (d) separately. The rasterized map in (b) is created from (a), where the drivable areas (in grey), lane markings (lane markings in light yellow, pedestrian crossings in white) and lane centrelines are filled with identical colours. The map graph in (d) is created from (c), showing the traversability of lane segments.

The main difference between multi-channel grid maps and RGB images is that the grid cells in grid maps are populated with numerical values instead of RGB values. This mechanism allows grid cells to store dynamic traffic contexts like traffic light states, which is not easy for RGB representation. Therefore, predefining orderings while stacking RGB images could be avoided.

In [53], road geometries and traffic rules, including traffic lights and lane types, are encoded by 17 binary maps in which the spatial grid cells are assigned with

1 or -1 if they are crossed by lane markings and ruled by traffic rules. This parametrization of the map allows the model to extract rich scene information from the geometric details together with traffic rule information. In [100], to enable variable receptive fields in the grid maps, single-channel grid maps with three different discretization resolutions are created to represent the scene. The three grid sizes are classified into 'local', 'medium' and 'global' to describe the scenario from micro trajectory features to the whole scenario context. In a grid map with a specific resolution type, grid cells are filled with predefined values representing the severity while conflicts happen to the occupied traffic agents, and the constraint intensity of traffic rules on vehicles. Cases of high severity of potential collisions and strong constraints of traffic rules are assigned with high numerical values. For example, motorcycles are given higher values than cars because they are more vulnerable while involved in an accident. Solid line markings have bigger values than dashed line markings because the solid ones are more restrictive.

The advantage of using image-like representation is that CNNs can efficiently and effectively abstract RL map features [78] in a broad spectrum of road scenes, ranging from simple highways to complex urban environments. Nevertheless, some limitations lie in road connectivity representation, topology-guided feature exploration and real-time application. First, image-like representation is not able to semantically describe lane connection types of lane pairs [88], for example, it is not easy to distinguish adjacent lane pairs in the same or opposite directions. Additionally, 2D convolution may not be able to capture global topology features among road lanes because of the limited receptive fields of the convolution kernel. This could be partly resolved by generating images with larger grid sizes [100], which might cover more global connectivity information in the convolution kernel. But the efficiency of this solution is still limited. Second, topology-guided feature exploration cannot be realized, like the feature extraction along a specific lane. Expansion on the reception field could cover more intended areas along a target lane but will inevitably contain more redundant regions, since the kernel treats the image pixels homogeneously. Lastly, the continuous discretization of

elements around the TA poses a challenge for real-time applications [54] [101] because it is time-consuming and should be conducted at each time step.

3.5.3.3 Graph Neural Networks

In autonomous driving, road-layout maps are usually served as prior. They are arranged with a set of lane segments and their connectivity in a vectorized way, providing structured representations of the scenarios with geometric and semantic pairs or with geometry entities alone. To better use this geographic and semantic information and preserve the map element connectivity, these vectorized road-layout maps are usually converted to a directed graph for further feature extraction, as shown in Figure 3-9 (d).

A primary map graph could be constructed based on actors' traversability across the connected lane nodes. These lane nodes usually consist of lane segments with similar lengths. A lane segment length of 10 to 20 meters could represent the roads on the map in a macro way, while a length of one meter or even smaller represents the roads in a micro way [86]. These lane segments often contain a sequence of ordered lane centrepoints [102]. Since these sequential centrepoints capture both the direction of the traffic flow and the legal routes that each driver can follow, their physical characteristics are often used as the lane node features [103].

3.5.3.3.1 Graph Nodes

The basic lane node is described by features inherited from lane centrepoints are their coordinates and orientations (e.g., yaw or tangent direction), capturing the geometry of the lane segments [103] [104] [105]. Semantic information like traffic control elements could also be added to the node features as a supplement to indicate whether the centrepoints lie on a stop line or crosswalk [103], to distinguish turning directions [104] and to show traffic light statuses [106]. To show the accessibility of a lane node, the spatio-temporal occupancy status of the lane segments is incorporated into the lane node features based on the historical trajectories of traffic agents [104].

Further, lane node features sometimes include traffic agents' information. In [106], apart from the above-mentioned geometric and semantic features, a lane node is matched with its nearby traffic agents, and these traffic agents' positions and orientations relative to the matched lane segment centres over their historical horizon are also added to the node feature. In this way, the lane node encodes the traffic agents' manoeuvre intention into its features, like approaching or leaving the matched lane node. In [103], lane node encodings are updated with nearby agent encodings using multi-head attention. In [105], spatial attention is used to capture actor-map interactions. Traffic agents within a spatial range of the selected lane node are extracted. The lane node features and traffic agent features, along with the relative position between the lane node and the actor node, are concatenated together and then projected into the new lane node feature. By this spatial attention, real-time traffic information is introduced to lane nodes, modelling road blockages and availability of the lanes.

Apart from the lane centrepoinets, vectors of polyline elements in the road-layout map could also be treated as graph nodes. In [101], a node is a polyline vector consisting of sequential points from lane boundaries (or motion trajectories). The node features are the start and end locations of each polyline, along with other attributes such as polyline ID and semantic labels.

3.5.3.3.2 Graph Edges

Individual lane nodes are linked by edges in the graph, representing the lane segment connectivity in the road-layout maps [107]. According to the adjacency of the lane segments in the map, edges have types of predecessors, successors, and left and right neighbours. Given a lane segment, its predecessor and successor are other lane segments that can directly travel to and from the selected segment. Left and right neighbours refer to the lane segments which can be directly reached without violating traffic rules in the traversal directions [105]. In [103], an idea of proximal edges is also used, which accounts for lane-changing manoeuvres between neighbouring lane segments. In this application, a distance threshold is used to identify the neighbour lane segments and their orientations

are checked to be within a yaw threshold of the ego lane segment to avoid erroneous assignment of nodes when multiple lanes overlap.

The edges are often stored in the form of an adjacency matrix. A value of one in the p row and q column represents that node p and node q have linkage, and vice versa. In [106] [105], four edge matrices instead of one edge matrix are used to store linkage information. Storing linkage information of all types of connections in separate matrices enables non-interfering exploration of individual connection types.

3.5.3.3.3 Graph

A map graph could be constructed in a global way by connecting all lane nodes in the scenario according to the map topology.

A fully connected graph is simple to construct but has high computational cost and it ignores specific semantic connections between elements.

To better explore local scenes and global topology, graph layers of both micro and macro levels are created simultaneously to encode information on a different scale. The global graph preserves the map structure and the local graph captures the more fine-grained information within a small region. In [104], two separate map graphs are established to focus on local traversability and global topological information separately. The nodes in the local graph are sampled points on the drivable area with a resolution of 1m around the target traffic agent, noted as the drivable area (DA) layer. And the global graph is created based on the lane segment centreline with a resolution of 2m, noted as the lane segments (LS) layer. The DA and LS layers are from the same map data and thus always spatially overlapped. To realize better local-global feature fusion, inter-layer edges between nodes of these two layers are established by matching nodes in the LS layer with their Euclidean neighbours in the DA layer. This connection enables message propagation forward and backward between these two layers. In [101], a hierarchical graph network is established. Based on the idea of constraining the connectivity of the graph based on the spatial and semantic proximity of the nodes, a subgraph is established with nodes in the same polyline

sharing the same semantic labels. Then, all local graphs are fully connected with each other to form a global graph for information exchange. In [106], a local graph named LaneRoI is built by nearby lane nodes of the TA and their connected nodes. Multiple LaneRols are then connected to form a global graph. This topology could better represent the local scenario context while being aware of global topology.

3.5.3.3.4 Message Propagation

The lane node feature above only captures the local information of a lane segment. To aggregate the topology information of the lane graph at a larger scale, the encoded lane nodes exchange information by message propagation through the graph, updating lane node encodings by collecting encoding info from its surroundings.

In some applications, message propagation among lane nodes is processed by pooling methods. In [101], message propagation is done by concatenating the target node encoding with the max-pooling results of its neighbour node encodings. In [106], different local graphs (LaneRoI) can include the same lane node for their spatial. To update the node feature, node features of all its neighbours, inside both its current local graph and other local graphs containing this node, are pooled based on MLPs. In [68], the features of the neighbour nodes are arranged in a stacked grid and a max-pooling operation is used to get the global feature inside the local DA graph. This max-pooled feature is then concatenated with the target node's feature and further processed by a fully-connected layer.

Pooling methods usually lead to a one-step aggregation from a node's direct neighbours. For a multi-step aggregation from indirect neighbours with an exploration depth in the graph topology, GCN could be used [86]. In [105], a graph convolution method is used to realize multi-scale message propagation along different directions of the lane nodes. Because traffic agents usually move farther in the lane longitudinal direction than the lateral direction over the prediction horizon, the graph convolution operates within four separate adjacency matrices to enable exploration in predecessors, successors, left neighbours, and right

neighbours with diverse depths. To accommodate the long-range dependency of lane node information, the graph convolution is further set to be expanded to enable exploration with customized depth along the lane edges, mimicking the long-range exploration ability of the dilated 2D convolution operator by enlarging the receptive field. In [106], a similar multi-scale anisotropic graph convolution method is built but it operates inside each LaneRoI, a local graph embedding both actor and map information.

In [104], GAT is used to realize message propagation according to the importance of the neighbours. The attention weights are calculated based on encodings of node features and the node aggregates neighbour encodings through weighted sum.

A summary of NNs for road scene mining is illustrated in Table 3-3.

3.5.4 Neural Networks for Fusion

Vehicles drive on the road, guided by the road geometries and layouts. These vehicles also have an influence on the dynamic states of the road, like the real-time traffic density. To form a comprehensive encoding vector which includes both vehicle and map information, the map encodings and agent encodings are commonly concatenated along a specific dimension. However, this simple concatenation does not well explore the intern relation between vehicles and maps, like the guidance of the road to the vehicle motion and evolution of traffic densities.

To provide a map-aware agent encoding or an agent-aware map encoding, these two heterogeneous encodings could be fused by exchanging features with each other, as shown in [105]. From the literature, the most commonly used methods include CNNs, AM, TF and GNNs.

3.5.4.1 Convolutional Neural Networks

CNN does the fusion based on a combination of representation of road scene and agent motion. With the rasterized road scene layers that have described the map elements [83] [96], traffic agents' dynamics are stacked additionally on top of the rasterized images. Each traffic agent is assigned a polygon to represent its

posture at the current time step and multiple polygons of the same colour but with reduced brightness levels to represent its historical states. Then CNNs are used to process these images uniformly [99] [108] to encode the spatial correlation between the agent and the map elements.

3.5.4.2 Attention Mechanism and Transformer

Both AMs and TFs explore the interrelationships among vehicles and road layout elements by cross-attention, in which a map-aware agent encoding or agent intention estimation is generated. For example, Ngiam et al. [109] have used a TF-based method to enable the agent features to be updated by attending to the road graph. In TF, the Q vectors are generated from agent encodings and K and V vectors are from the road graph embeddings. The map-aware agent encoding is calculated by cross-attention based on these Q, K and V vectors. Conversely, in some other research works [86] [110] [75], vehicle information is projected into the lane node feature with similar approaches. For example, Guo et al. [111] map the Q vector from the centreline node feature encoding while obtaining K and V vectors from the vehicles' encodings. And then multi-head attention is used to calculate the weights of influence of different vehicles on the lane node, after which vehicle encodings are allocated by weight production. In that way, the traffic flow density could be inferred in the model. A stacked structure is used in [68] to incorporate multiple channels of agent motion and traffic scene data. More specifically, three TFs are arranged in a hierarchy to produce an initial prediction trajectory proposal and fuse map and interaction information step by step.

AMs and TFs can also identify potential succeeding lane segments in the prediction horizon. For lane selection, Luo et al. [112] have fused the embeddings of the historical trajectory of the TA and centrelines of potential proceeding lanes to predict the probability of each potential lane that the vehicle might head for. The historical trajectory is encoded by LSTM while the centrelines are encoded by 1D convolution for efficiency. Then dot-product and a Softmax layer are used to calculate the final probabilities of each potential lane.

Table 3-3 A summary of road scene mining architectures

NNs	Inputs	Works	Pros	Cons
Implicit	Vehicles' states relative to road elements	[57] [95]	<ul style="list-style-type: none"> • Simple in input data arrangement • No extra NNs needed to process road scene data • Efficient in simple road scenes like highways 	<ul style="list-style-type: none"> • Lack of awareness of the global road scene
GNN	Lane node features and adjacent matrices	[86] [105] [101] [102] [103] [104] [106] [107]	<ul style="list-style-type: none"> • Able to construct a heterogenous graph to contain both geometry and semantic information • Able to realise topology-guided feature exploration, like the feature extraction along a specific lane • Reduce time consumption because static road layout can be pre-processed and reused in the stage of training • Enable multi-step message passing with customized depth • Flexible to explore both local and global scenarios 	<ul style="list-style-type: none"> • Time-consuming and storage-inefficient to construct and store a dynamic graph at each time step
CNN	Raterised images	[83] [96] [100] [99]	<ul style="list-style-type: none"> • Efficient and effective abstraction of map features • Uniform representation of traffic scene context like surrounding vehicles and road lane markings • Highly adaptive to a broad spectrum of road scenes, ranging from simple highways to complex urban environments 	<ul style="list-style-type: none"> • Time-consuming in scenario rasterization • Have information loss because of spatial occlusion of different elements • Inefficient in capturing global topology features because of the limited receptive fields of the convolution kernel • Not able to perform topology-guided feature exploration

Grid maps	[52] [100]	<ul style="list-style-type: none"> • Efficient and effective abstraction of map features • Allow to store traffic contexts like traffic light states • Avoid predefining orderings of different traffic scene context • More informative because channel layers can be customised • Highly adaptive to a broad spectrum of road scenes, ranging from simple highways to complex urban environments 	<ul style="list-style-type: none"> • Time-consuming in scenario discretization • Inefficient in capturing global topology features because of the limited receptive fields of the convolution kernel • Not able to perform topology-guided feature exploration • Occupy more storage than RGB images
-----------	------------	---	--

3.5.4.3 Graph Neural Networks

While GNN techniques are used, map elements are usually arranged as nodes in the graph structure. The information exchange between vehicles and map elements could be operated by integrating vehicle information into the map nodes first and then propagating through the whole map graph.

Liang et al. [105] aggregate vehicle encodings to the lane node features to capture the real-time traffic information and then GCN is used to propagate this fused information over the lane graph. This approach is also applied in [111] and [104].

Alternatively, vehicles could also act as nodes in the graph to engage in message propagation directly with lane nodes.

Mo et al. [113] have used GAT to fuse vehicle and map information to model the effects of a TA's neighbours and centrelines on its future motion. A heterogeneous hierarchical graph containing both traffic agents and lane centrelines is created. These two types of node features are fused step by step. Three separate GATs are used to gather information from centrelines to the SAs, from the SAs to the TA, and finally from centrelines to the TA. By this sequential fusion, the TA obtains information both from SAs and road layouts.

Tang et al. [81] accumulate information from the TA's neighbourhood to the TA by GCN at each historical time stamp. The TA's neighbourhood information includes both SAs' historical trajectories and lane border segments. By message passing through both nodes of SAs and map elements, the TA generates a comprehensive neighbourhood context encoding. This dynamic graph is also used in [114]. A comprehensive adjacency matrix is created by extending the vehicle-vehicle adjacency matrix with road network connection information through a weighting factor, which determines the influence of the road network topology on the vehicle interaction graph.

A summary of NNs for traffic scene context fusion is illustrated in Table 3-4.

Table 3-4 A summary of fusion architectures

NNs	Works	Pros	Cons
CNN	[83] [98] [3] [108]	<ul style="list-style-type: none"> • Efficient in feature extraction operation • Well explore the spatial relation between map elements and agents 	<ul style="list-style-type: none"> • Time-consuming and storage-intensive in scenario rasterization • Hard to extract semantic constraints of the map elements on the vehicle motions
AM & TF	[86] [109] [111] [110] [112]	<ul style="list-style-type: none"> • Allow information fusion among heterogeneous scenario encodings parallely • Powerful in interaction extraction • Aggregate interaction information according to importance 	<ul style="list-style-type: none"> • Burden in cached storage while the sizes of the attended elements and their features are large
GNN	[81] [105] [104] [111] [113] [114]	<ul style="list-style-type: none"> • Enable direct information exchange by building a heterogenous graph • Provide customized connections among traffic scene elements, such as priority-aware and distance-aware connections 	<ul style="list-style-type: none"> • Time-consuming and storage-intensive to construct a dynamic graph at each time step

3.6 Performance Comparison

This section first introduces some commonly used quantitative evaluation metrics. Then the effectiveness of some state-of-the-art prediction models is compared.

3.6.1 Evaluation Metric

Commonly used displacement error metrics for single-modal trajectory prediction include Final Displacement Error, Average Displacement Error, Root Mean Square Error, and Mean Absolute Error.

Final Displacement Error (FDE) [15]: This metric calculates the L2 distance between the predicted trajectory end point and the ground truth end point:

$$FDE = \sqrt{(x_{t+p} - x_{t+p}^{gt})^2 + (y_{t+p} - y_{t+p}^{gt})^2}, \quad (3-3)$$

where x_{t+p} , y_{t+p} represent the predicted 2D position coordinates of a TA at the last time step ($t + p$) of the prediction horizon. The superscript gt represents that the data are from the ground truth. The final trajectory prediction point is related to the TA's driving intention, so it is important to evaluate its accuracy [15].

Average Displacement Error (ADE) [15]: It is used to measure the overall distance deviation of the predicted trajectory to the ground truth:

$$ADE = \frac{1}{p} \sum_{i=1}^p \sqrt{(x_{t+i} - x_{t+i}^{gt})^2 + (y_{t+i} - y_{t+i}^{gt})^2}, \quad (3-4)$$

where x_t , y_t represent the predicted trajectory point at the timestep $t + i$. The same variables are used in later formulas.

Root Mean Square Error (RMSE): This metric is sensitive to large prediction errors and is one of the commonly used metrics for trajectory prediction. It is the average sum of squared distances of total timesteps from a single predicted trajectory to the ground truth:

$$RMSE = \sqrt{\frac{1}{p} \sum_{i=1}^p [(x_{t+i} - x_{t+i}^{gt})^2 + (y_{t+i} - y_{t+i}^{gt})^2]}. \quad (3-5)$$

Mean Absolute Error (MAE) [15]: It is the average absolute error between the predicted trajectory and the ground truth t , and is generally used in the horizontal and vertical directions:

$$MAE_x = \frac{1}{T_{pred}} \sum_{t=1}^{T_{pred}} |x_t - x_t^{gt}|, \quad (3-6)$$

$$MAE_y = \frac{1}{T_{pred}} \sum_{t=1}^{T_{pred}} |y_t - y_t^{gt}|. \quad (3-7)$$

The evaluation metrics commonly used for multimodal trajectory prediction include minFDE-K, minADE-K, where K denotes the total number of modalities of the predicted trajectories. minFDE-K means the minimum FDE among K predicted trajectories. In Argoverse [8], minADE-K corresponds to the ADE of the predicted trajectory with the minimum FDE value, which aims to ensure the consistency of the optimal predicted trajectory.

Miss Rate (MR) [15]: If the predicted trajectory's endpoint deviation from the true endpoint exceeds a threshold d for a sample, that sample is called a miss. The ratio of the total number of misses m to all samples M is called the miss rate. MR reflects the overall closeness of the model's prediction of the trajectory endpoint. In Argoverse [8], a predefined fix threshold d of 2 m is used.

3.6.2 Comparison of State-of-the-art Methods

The training and validation of DL-based prediction algorithms are dataset-dependent. Different datasets may have distinct evaluation metrics, making it difficult to directly compare all methods. This subsection will compare some state-of-the-art methods based on two frequently used datasets: NGSIM and Argoverse as presented in Table 3-5 and Table 3-6. The data in the tables are all from the publications and model parameters are fine-tuned by models' creators to achieve competitive performance.

Table 3-5 includes the RMSE of trajectory prediction across various prediction horizon values. The results are sorted by ascending the final prediction error (at 5 s). It can be seen that the larger the prediction horizon, the lower the prediction accuracy (lower RMSE values, better the prediction performance). Regarding the prediction backbones, LSTMs are popular. They are commonly used in encoding historical states of the TA, as well as those of SAs. The produced encodings of SAs are then permuted and used in interaction mining. Though the pipeline is similar across different interaction mining methods, the details of how SAs are identified and arranged vary. CS-LSTM [1] selects most adjacent SAs and arranges them in a grid centred on the TA. This TA-centric interaction mining narrows down the field of interest (FOI) to explore potential interaction and turns out to be more effective than identifying all vehicles in the current scene, as used in MATF [91]. Prediction accuracy is further improved when SAs are arranged in non-Euclidean ways, like a graph [77] or in polar coordinates [71] [115]. These topologies better identify SAs and are more adaptive to different road geometry like, for example, a curve. Advanced NNs like GNNs [77] and AMs [115] [116] can be used for interaction mining. In HLTP [116], a combination of NNs is used to mine the agent interaction. An AM-based method is used to explore the short-term interaction with neighbours in the forward vision while another CNN-based method is used to mine the interaction of all SAs. Then these two encodings are fused by TFs to generate a comprehensive encoding of the traffic context. Through Table 3-5, it can be seen that explicit interaction mining has proven to provide more accurate trajectory prediction results than implicit mining, which is used in S-LSTM [56] and D-LSTM [57], because NNs have limited ability to mine interaction directly from tedious input matrices consisting of original states of both the TA and SAs.

Despite the advancements in the organisation of SAs, the development of the prediction backbone is crucial to improve the prediction accuracy. In the applications of LSTM, the sequential structure of the encoder-decoder helps reduce the prediction errors. Table 3-5 shows that, though implicit interaction mining is used, M-LSTM [60] has an encoder-decoder-based model and achieves better results than S-LSTM [56] and D-LSTM [57] which do not have an

equivalent LSTM decoder. Further improvements are achieved by applying AM and TF in the prediction backbone. In IH-Net [90], the prediction backbone is still an LSTM-based encoder-decoder. Differently from the widely applied encoder architecture that uses the hidden state of the last LSTM cell as the encoder's final output, hidden states at all time steps are cached in a feature matrix. In the decoding stage, additional high-dimension features are obtained by attending the hidden state of the decoder LSTM cell to the feature matrix by cross attention. This feature encoding is then concatenated with the decoder LSTM hidden state to predict the future position. This combination of LSTM and AM improves the ability to extract historical features and makes IH-Net rank first in the NGSIM dataset considering its RMSE of 1.86m at the final prediction time stamp.

In NGSIM, although road scene information like the corresponding driving lane of vehicles and deviations to the lane markings are provided, road layout information is seldom used for lack of a fine-grained road layout map.

Table 3-6 compares some state-of-the-art methods based on the Argoverse 1 Motion Forecast Dataset [8]. The Argoverse dataset provides varying traffic scenes in urban scenarios. An accurate trajectory prediction relies on a comprehensive understanding of the traffic scene and a good handling of manoeuvre uncertainties. The former challenge calls for powerful methodologies on agent interaction and road scene mining while the latter is usually resolved by a multi-modal prediction.

In Argoverse, the observation horizon and prediction horizon are 2 s and 3 s, respectively. The selected metrics are minADE, minFDE, and MR (the distance threshold of MR is 2 m). Different methods are sorted in ascending order by minADE (K=6), where a smaller value indicates a better result in the multi-modal prediction. If the minADE (K=6) does not exist, the models are sorted by minADE (K=1) instead. In addition, the numbers of trainable parameters of some prediction models are listed. These values are collected from [101] [117] [118] [119] [120].

Argoverse samples anchor points across the whole drivable area (DA) in a dense and even-distributed way. Benefiting from these prior points, it is convenient to

relate the dynamic and static traffic scene elements to these points and create a uniform discrete representation in rasterised RGB images or multi-channel grid maps. CNNs are usually used as backbones to predict the multi-modal predictions in these cases.

Despite the efficiency of CNNs, the rasterization of traffic scenes can be time-consuming, potentially hindering real-time applications. To address this issue, VectorNet [101] introduces a novel approach by representing agent trajectories and lane boundaries as sparse polyline elements. It leverages GNNs to extract high-dimensional features within each polyline, while AM is applied to capture interactions among all agent and lane boundary polylines. VectorNet achieves a minADE of 1.810m and a minFDE of 4.010m. Although its performance does not surpass CNN-based models like HOME, which has a minADE of 1.690m, a minFDE of 3.650m, and a parameter count of 5100K, VectorNet stands out with its lightweight model of only 72K parameters and its capability to explore both local and global road scenes. Consequently, it is still widely used in other methods as a preprocessing step for road scene information [68] [121].

LaneGCN [105] builds on VectorNet's concept by representing lane segments using their centrelines and constructing lane graphs based on the connection types of these centrelines. Unlike VectorNet, which fuses agent interactions and road scene information in a single step by creating a heterogeneous graph containing both lane and agent nodes, LaneGCN introduces a map-aware agent encoding process that is performed step by step. First, agents' encodings are aggregated into their surrounding lane nodes using AM. Next, GCN propagates information among the connected lane nodes. Finally, the agent encodings are updated by aggregating the lane node features back to the agents through AM. This fusion process enables agents to better perceive the traffic and geometric characteristics of lanes, leading to a more accurate understanding of traffic scenarios, as evidenced by a minADE of 1.710m and a minFDE of 3.780m. This hierarchical fusion approach is also employed by models like LaneRCNN [106], DSP [104], and HeteroGCN [122]. In these methods, for enhanced local exploration, lane nodes surrounding the agent are retrieved to form a local graph

and GNNs are used to propagate information among both agent and lane nodes. Subsequently, a global graph is constructed to facilitate interaction mining on a broader scale.

From Table 3-6, it can be seen that top-performing models surpass the others by incorporating AM and TF technologies, particularly excelling in multi-modal evaluations. Generally, these models achieve reliable multi-modal predictions by equipping TF decoders with learnable queries to generate predictions, drawing inspiration from the TF-based object detection method DETR [123]. These learnable queries are updated through the process of interaction mining and road scene mining, thus capturing the traffic scene context.

mmTransformer [68] is among the first models to apply TFs. It utilizes three stacked TFs to extract historical motion features of agents, aggregate global static map information, and mine agent-agent interactions. Within each TF, the learnable decoder queries are updated with the output of the TF encoder, gradually incorporating traffic scene features. This strategy turns out to be efficient and effective, outperforming GNN-based and CNN-based solutions with a minADE (K=6) of 0.840m and minADE (K=1) of 1.360m. The performance of scene understanding is further enhanced by focusing on the neighbourhood of the TA and then integrating this local information with a global operator, as demonstrated in HiVT [119]. In this approach, a local region is identified around each agent, where agent-agent and agent-lane interactions are explored using AM at each time step. These local features are then integrated and processed through self-attention to learn spatial interactions among agents.

Advancing this concept, a more comprehensive approach involves combining all map encodings and agent encodings and applying self-attention to simultaneously capture interactions and road scene context. This method has proven more effective in traffic scene analysis, as seen in models like ProphNet [118], Wayformer [124] and SEPT [121]. SEPT [121], which ranks first in Table 3-6, further enhances its ability to capture agent motion and road scene features through self-supervised learning. Specifically, its agent motion and road scene encoders are pre-trained before the trajectory prediction training stage to

reconstruct partially masked trajectory and road border line samples by self-supervised learning, in which smaller deviations from the reconstructed sample to the ground truth indicate a deeper understanding of the traffic scenes. Then, these pre-trained encoders are trained together with the decoder in the motion prediction training stage.

Considering the comprehensive reasoning abilities of AM and TF, there appears to be a trend to replace the GNN-based message propagation mechanism with the attention mechanism, though vehicles and road layout elements are still represented in a graph. Another factor that leads to the popularity of the AM and TF is the success of their deployment in identifying potential succeeding lane segments in the prediction horizon. These estimations better model the multimodal intentions inherent in road driving, because they follow the guidance of road layouts and traffic rules, compared to other approaches that rely on random sampling as GFs do. It can be seen from Table 3-6 that AM and TF dominated models, such as LAformer [117], HiVT [119], and mfTransformer [120], have a smaller size of model parameters compared to CNN and GCN dominated ones, like HOME [83] and LaneGCN [105]. This highlights the advantages of AM and TF in compactness and capacity in trajectory prediction tasks.

Table 3-5 Comparison of state-of-the-art model performance based on NGSIM

Literature	Backbone	Interaction mining	Scene mining	Fusion	RMSE				
					1s	2s	3s	4s	5s
IH-Net [90]	LSTM, AM	CNN	-	-	0.45	0.81	1.11	1.50	1.86
HLTP [116]	LSTM	AM, CNN, TF	-	-	0.48	0.99	1.54	2.22	3.09
GRIP [77]	LSTM	CNN, GNN	-	-	0.64	1.13	1.80	2.62	3.60
BAT [115]	LSTM	AM	-	-	0.23	0.81	1.54	2.52	3.62
Polar-V [71]	LSTM	Pooling	-	-	0.25	0.84	1.58	2.53	3.75
ST-LSTM [72]	LSTM	LSTM	-	-	0.56	1.19	1.93	2.78	3.76
CS-LSTM [1]	LSTM	CNN	-	-	0.61	1.27	2.09	3.10	4.37
M-LSTM [60]	LSTM	Implicit	-	-	0.58	1.26	2.12	3.24	4.66
MATF [91]	LSTM	CNN	CNN	Concat.	0.67	1.51	2.51	3.71	5.12
D-LSTM [57]	LSTM	Implicit	Implicit	-	0.49	1.41	2.60	4.06	5.79
S-LSTM [56]	LSTM, MLP	Implicit	Implicit	-	0.72	2.00	3.76	5.97	9.01

Table 3-6 Comparison of DL-based model performance on Argoverse

Literature	Backbone	Interaction mining	Scene mining	Fusion	K = 1			K = 6			Param
					min	min	MR	min	min	MR	
					ADE	FDE		ADE	FDE		
SEPT [121]	TF, MLP	AM	AM	AM	1.441	3.178	0.515	0.728	1.057	0.103	-
QCNet [125]	AM	AM	AM	AM	-	-	-	0.730	1.070	0.110	-
Wayformer [124]	TF	AM	AM	AM	-	-	-	0.767	1.163	0.119	-
LAformer [117]	GRU	AM	GRU	AM	-	-	-	0.770	1.16	-	2654K
ProphNet [118]	TF	MLP	MLP	AM	1.524	3.334		0.773	1.144	0.112	-
HiVT [119]	MLP	AM	MLP	AM, MLP	-	-	-	0.774	1.169	0.127	2529K
Multipath++ [126]	LSTM, MLP	LSTM, AM	MLP	AM	-	-	-	0.790	1.214	0.132	-
mfTransformer [120]	TF	TF	TF	TF	-	-	-	0.819	1.275	0.143	2469K
HVTD [81]	TF	GCN	GCN	AM	1.590	3.620	0.580	0.820	1.270	0.150	892K

mmTransformer [120]	TF	TF	GNN	TF	1.770	3.780	0.591	0.840	1.340	0.150	2607K
LaneGCN [105]	MLP	AM	GCN	FC, GCN	1.710	3.780	0.591	0.870	1.360	0.160	3701K
LaneRCNN [106]	MLP	GCN	GCN	GCN	1.690	3.690	0.569	0.900	1.450	0.120	-
HOME [83]	GRU, MLP	AM	CNN	-	1.690	3.650	0.571	0.940	1.450	0.100	5100K
LaPred [102]	MLP, LSTM	FC	LSTM	FC	1.480	3.290	-	-	-	-	-
HeteroGCN [122]	MLP, GRU	GCN	GCN	GCN	1.570	3.410	0.540	-	-	-	-
SSL-Lanes [107]	MLP	AM	GCN	AM	1.630	3.560	0.567	-	-	-	-
DSP [104]	MLP, CNN	GAT	GCN	GAT	1.687	3.710	0.575	-	-	-	-
VectorNet [101]	MLP	GCN	MLP	GCN	1.810	4.010	-	-	-	-	72K

3.6.3 Discussions of Future Research Directions

The following are some of the research gaps in vehicle trajectory prediction and could be considered as opportunities for future works:

1. Time efficiency: most of the existing works in the literature mainly focus on the improvement of prediction accuracy. The complexity surges because a single NN is stacked multiple times for dedicated feature mining and different types of NNs are integrated for heterogeneous data processing, e.g., image data and numerical data. Those can potentially propose challenges in real-time applications. Thus, in model designing, runtime performance characteristics of NNs should be taken into consideration, e.g., if parallel computation is supported. And a metric of computational burden should be included during the model evaluation to boost the applicability of trajectory prediction.
2. Storage requirement: the operation of the trajectory prediction models relies on both dynamic and static memories. The intermediate outputs between two sub-modules and dynamic sequential data, like historical trajectory data, consume dynamic memories. At the same time, road scene data, like road layout data, may occupy static memories. Thus, it is worth studying prediction models with light and integrated structures to enable fewer intermediate outputs. The application of sparsely represented road layout data should be explored because, despite some limitations in accurate scene representation, it enables efficient vehicle-map matching and minimal storage requirements.
3. Model generalization: Currently, trajectory prediction models are trained in a small variety of scenarios, like several roundabouts and cross intersections, limited by the availability of the training dataset. Also, some data imbalance exists in these datasets. For example, going-through trajectories might dominate the manoeuvres while few turning trajectories exist. These two issues result in poor performances when the models are exposed to scenarios with different geometries, topologies and traffic

characteristics. So, the applicability of the proposed models to different scenarios should be carefully evaluated.

4. Accurate prediction of heterogeneous traffic agents: Most datasets only provide monotonous scenarios, with either vehicle or vulnerable road users dominating the traffic scenarios. Based on these datasets, a trained model might not perform well while predicting trajectories of heterogeneous traffic agents, since they might have significantly different motion patterns. A prediction model that enables an accurate prediction of all types of traffic agents will improve the prediction and the safety of AVs, since heterogeneous traffic is very common, especially in urban areas.
5. Robustness and turbulence tolerance: Complete observability of all traffic agents and smooth tracking are assumed in most of the available datasets. This is not the case in many real-world applications, because measurement noises and occlusions are important factors to consider from the perspective of onboard sensors. The performance of prediction models should therefore be assessed while considering these factors.

3.7 Conclusion

The DL-based methods used in vehicle trajectory prediction were reviewed by organizing them into three categories: trajectory prediction backbone, interaction mining, and road scene mining. In the prediction backbone, vehicle trajectory prediction highly depends on vehicle historical states, which are usually obtained by stacking historical data across the temporal dimension. For the representation of vehicle interactions and road scenes, relevant data are arranged in specific structures, like graphs and grids. These data are sometimes further converted into formats that are suited for the operations of NNs, typically to encode temporal or spatial features. For example, vector maps are often converted to rasterized images for convenience of CNN processing. This analysis revealed why a specific NN architecture is chosen according to the component scope and the input data type and how similar NN architectures could be used in very different components of the prediction model, with different scopes and input structures. Furthermore, the performance of several prediction models was compared according to some

commonly used evaluation metrics. It was found that appropriate encodings of vehicle interaction and road scenes improve the prediction accuracy because they provide a better understanding of the driving scenario. In addition, the results showed that AMs have the ability to mine both vehicle interaction and road scenes, achieving state-of-the-art prediction performance compared to other NNs. At last, some potential research directions were outlined in this chapter, mainly focusing on the practical application of vehicle trajectory prediction models.

4 Framework Methodology

Through the analysis of the accessible naturalistic driving (ND) dataset (Section 2.1), road layout (RL) representation for traffic scenarios (Section 2.2) and the review of deep-learning-based vehicle trajectory prediction, it is possible to argue that trajectory prediction research is generally conducted on a specific scenario without map information, or combined with agent-centric map information which is updated as the agents proceed. These researches have a limitation on a deep exploration of the influence of RL information on the trajectory prediction tasks for either lack of map data or too-fast-changing RLs, especially in the urban scenarios. This project explores the influence of static RLs on trajectory prediction in urban scenarios. To include various RL topologies, three urban scenarios are explored. This project's methodology involves pre-processing dataset data of RL map and traffic agent trajectories, and developing trajectory prediction models considering agent interaction mining and road scene mining.

While the article-based Chapters 5 to 8 cover most of the methodology of this project, they do not present it in relation to the entire scope of this thesis. Thus, this chapter consolidates the thesis methodology from dataset selection to dataset processing, and trajectory prediction method development. Research methods in this project can be summarised as follows:

1. **Dataset selection:** There exist several datasets containing naturalistic driving trajectories. Most relevant datasets targeting urban scenarios are picked up and compared to choose the most suitable one. The description of this part is shown in Section 4.1.
2. **RL data processing:** RL data of Lanelet2 formats are contained in the selected dataset. While the map files provide detailed geometry and semantic information, these data are not readily accessible on our research platform, neither MATLAB nor Python. Therefore, efforts are made to reformat these RL data. The first task is to check the original map data and fix the errors existing in the maps. Then, fundamental map information is extracted from the original files, as described in Section 4.4.2. A novel maximal-disk-based lane centreline extraction method is proposed to calculate the lane centrelines,

which are used as a basis of road scene mining in the trajectory prediction task but not provided as a prerequisite in the map data. The details of the centreline extraction methods are in Chapter 5.

3. **Trajectory data processing:** This project develops deep-learning-based (DL-based) trajectory prediction models based on raw data from a large-scale open-source ND dataset. Though this dataset is partly aimed at data-driven applications, the raw data cannot be fed into DL-based models for training. Trajectory clustering, pruning and splitting are performed for model training purposes and can be found in Section 4.4.3. For the preparation of agent interaction mining and road scene mining, surrounding agents (SAs) of the target agent (TA) are identified and the agent statuses are matched with the RL map elements. These two parts can be found in Section 4.4.4.1 and Section 4.4.4.2.
4. **Trajectory prediction backbone model:** this project first investigates the backbone of the trajectory prediction model which makes trajectory predictions based on agent historical trajectory encoding. An encoder-decoder prediction structure based on Recurrent Neural Network (RNN) units is selected as the backbone due to its efficiency in time-series prediction tasks and adaptive prediction horizon. Different RNN units (LSTM and GRU) and hyper-parameters are selected and compared to investigate their influence on prediction accuracy. The details can be found in Chapter 6.
5. **Trajectory prediction model considering agent interaction mining:** the backbone model is further extended to perform trajectory prediction considering the results of agent interaction mining. A reference design, based on an LSTM encoder-decoder structure, is selected from the literature and experiments are performed to investigate better solutions to concatenate the agent historical trajectory encoding and interaction encoding, through little modification of the reference design. The used methods are described in Chapter 7. This modified model gives some insights on how to integrate RL encodings in the development of map-aware prediction models.
6. **Trajectory prediction model considering road scene mining:** a map-aware prediction model is established which takes advantage of RL

information. Map elements are extracted from the sparsely described RL maps and are represented by their centrelines. 1D convolution (Conv) and Attention Mechanism (AM) are used to encode the map elements and create agent-map attention. Details have been presented in Chapter 8.

4.1 Dataset Comparison and Selection

The objective of this project is to predict the trajectories of high-mobility traffic agents in urban traffic environments. Trajectories of the target agent (TA), alongside its surrounding agents (SAs), will be analysed and investigated. Consequently, the dataset used in this research must encapsulate traffic agents with minimal occlusion in detection, to ensure accurate analysis.

Additionally, the research will delve into the role of geographic road layout (RL) information in influencing trajectory patterns. Therefore, it is imperative that the chosen dataset either inherently includes detailed map information or allows for the straightforward extraction of such data on the target region.

Moreover, the dataset should be comprehensive, encompassing sufficient recordings from one or multiple urban scenarios with a fixed recording view perspective. This amount of data is crucial to facilitate a thorough investigation into the correlation between trajectory patterns and the specific layout of roads and urban infrastructure with similar features at the same location.

Finally, to gain insights into real-world driving behaviours, the project will place emphasis on naturalistic driving (ND) data. The data recording ought to take place with no perception of traffic agents to minimize the influence on their driving behaviours. In this case, datasets that offer a bird's-eye view perspective are preferred, as they provide a comprehensive vantage point for observing and analysing traffic movement and interactions within the urban landscape.

Considering the above requirements, the most relevant datasets from the literature review are NGSIM [13], INTERACTION [17] and inD [20].

These datasets are all collected by cameras hovered at high attitudes, which are merely perceived by the traffic agents and thus will not influence their driving

behaviours. And they all provide a bird-eye-view (BEV) recording of the whole scenario. Owing to this ideal viewing angle, an entire intersection scenario can be measured with significantly less occlusion than with sensors at ground level [6]. In addition, they all provide map information that allows direct extraction of the geometry of the traffic scenes.

By leveraging the advantages and disadvantages, the inD dataset was chosen as the most suitable for this research project. The detailed considerations are listed below.

4.1.1 Sufficient Data

The inD dataset contains a total of four intersections and provides sufficient data in terms of trajectory amounts and trajectory variation. It includes trajectories of more than 11,500 road users, covering cars, trucks, busses and vulnerable road users (VRUs) such as pedestrians and bicyclists.

4.1.2 High Position Accuracy

The positional error in the inD is typically less than 10 centimetres [127], like what they have achieved in the highD.

This outperforms the position accuracy in the NGSIM. In NGSIM, two urban scenarios are recorded. As noted in previous works [13], [128], raw NGSIM trajectories are unsuitable for further analysis. The bounding boxes in the NGSIM dataset rarely align with vehicle shapes, and several outliers mostly contain road surfaces [13], as shown in Figure 4-1. These inaccuracies, especially at transitions between adjacent cameras that record consecutive segments, are caused by image stitching, leading to unrealistic speeds and accelerations.

The tracking system errors in the NGSIM dataset cannot be corrected by filtering [13]. This results in false positive trajectory collisions and physically unrealistic vehicle speeds and accelerations [6]. To address these issues, [128] smoothed the vehicles' longitudinal movements based on the original trajectories and released an updated version, excluding infeasible trajectories. However, these improvements are limited to a small portion of the dataset. Additionally, [13]

indicates that this method is not sufficient in all cases, requiring vehicles to be manually re-extracted from the recordings to achieve improved longitudinal trajectories. These could also be spotted in the urban scenarios data.

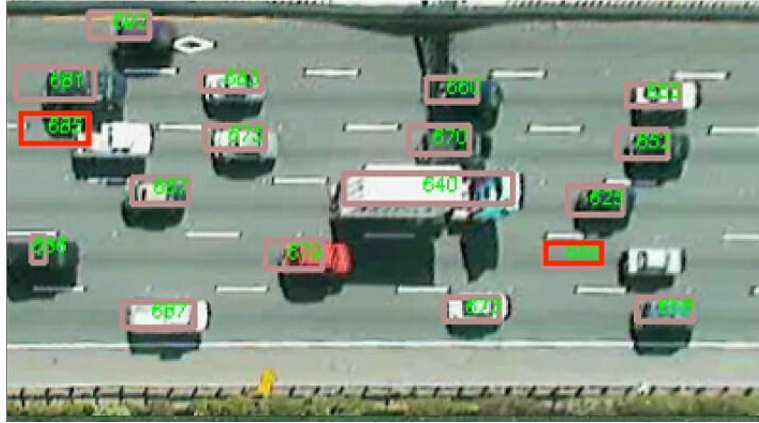


Figure 4-1 The mapping between bounding boxes and vehicle shapes is poor in some cases. Some bounding boxes do not overlap with the vehicles and contain the road surface (in red) [13].

4.1.3 Comprehensive Interaction

Focusing on the urban scenarios, the inD provides a more representative distribution of road user types and pays more attention to interactions between vehicles and between vehicles and other VRUs, like bicyclists and pedestrians.

The INTERACTION includes 3 unsignalized intersections and 1 signalized intersection. These two types of scenarios include 14,867 and 3,775 vehicles separately. An illustration of the recorded scenarios is shown in Figure 4-2. This dataset offers a high density of traffic and highlights highly interactive and complex driving behaviours, featuring both cooperative and adversarial vehicle motions.

However, only vehicle trajectories are recorded in the unsignalized intersections. This causes some bias in studying the interaction among urban traffic participants for usually urban intersections are shared by both vehicles and vulnerable road users, like pedestrians and cyclists.

At signalized intersections, traffic participants are mostly controlled by traffic lights, resulting in rare interactions.

Finally, the complexity of the road layouts of included intersections is high, which is not suitable for an easy start of trajectory prediction research.

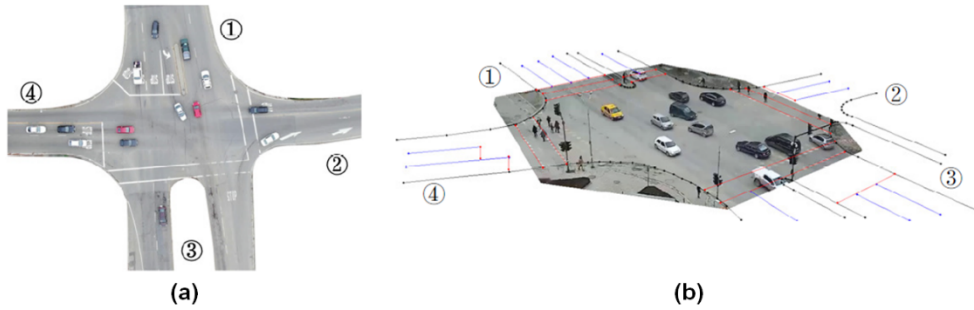


Figure 4-2 Examples of intersections recorded in the INTERACTION [17].

In summary, the inD is much more relevant for our research purposes considering the data sufficiency, position accuracy, and agent-to-agent interaction.

4.2 Introduction of Intersection Drone Dataset (inD)

4.2.1 An Overall View

In the inD dataset, trajectories are extracted from recorded videos captured by a drone in 4K (4096x2160 pixels). Each flight records approximately 20-22 minutes of video at a resolution of 25 Hz with maximum video quality. The class and track data of each road user are then extracted with high precision through deep neural networks.

In the dataset, not only the trajectory information is provided, but also some actor dynamic information and labelling information. Data fields relevant to our research are listed in Table 4-1 below:

Also, the class ID of each road user is given. A total of four classes are included. The first is for cars only. The second class contains buses and trucks. The third one is bicyclists, which includes motorbikes as well as bicycles. The last one is the class of pedestrians.

Table 4-1 Time-dependent variables for tracks in the inD

Name	Description	Unit
trackId	The ID for the track	-
frame	The frame for which the information is given	-
xCentre	The x position of the object's centroid in the local coordinate system, with a different variable, xUtmOrigin providing UTM coordinates	m
yCentre	The y position of the object's centroid in the local coordinate system, with a different variable, xUtmOrigin, providing UTM coordinates	m
heading	The heading in the local coordinate system	deg
width	The width of the tracked object	m
length	The length of the tracked object	m
xVelocity	The velocity in x-axis direction in the local coordinate system	m/s
yVelocity	The velocity in y-axis direction in the local coordinate system	m/s
xAcceleration	The acceleration in x-axis direction in the local coordinate system	m/s ²
yAcceleration	The acceleration in y-axis direction in the local coordinate system	m/s ²
lonVelocity	The longitudinal velocity	m/s
laTAelocity	The lateral velocity	m/s
lonAcceleration	The longitudinal acceleration	m/s ²
latAcceleration	The lateral acceleration	m/s ²

The inD recordings took place at four different locations with urban intersections in Aachen, Germany, from 2017 to 2019. As shown in Figure 4-3, all four intersections are unsignalized, with a speed limit of 50 km/h and walkways present. The measurement locations vary in intersection shape, lane numbers and types, right-of-way rules, road user types, and interaction type [20].

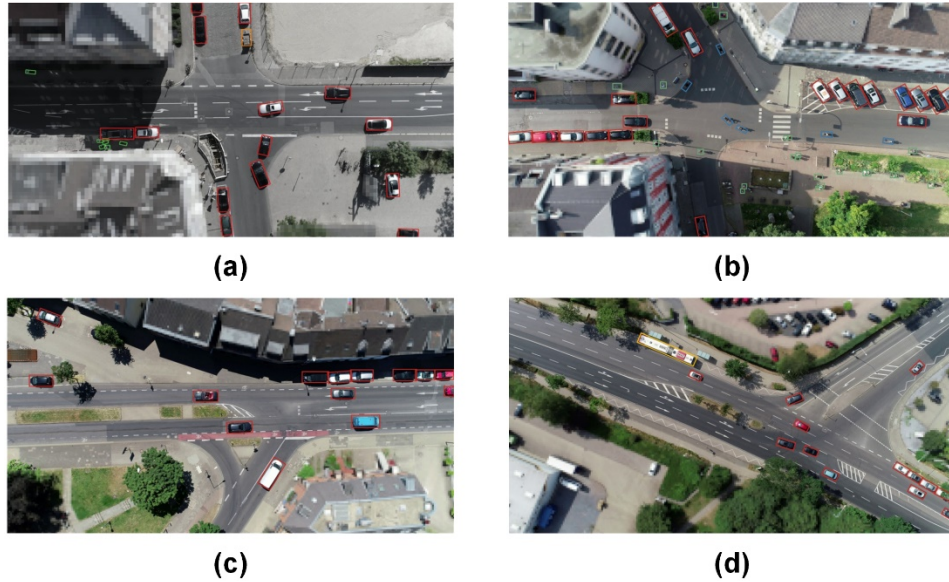


Figure 4-3 Four target scenarios in the inD [20].

4.2.2 Separate Scenarios

According to the descriptions in [20], the features of four intersection scenarios are summarised in Table 4-2. The numbers of recorded road users are listed in Table 4-3.

From the data, Bendplatz provides the most trajectories of vehicles. The ratio of cars to VRUs is appropriate, which provides potential interaction information between vehicles and pedestrians. Heckstrasse provides the least number of vehicle tracks. In Neukoellner Strasse, the pedestrians only take up a very small part of the data. Frankenburg presents unique challenges. It includes numerous parking spaces, resulting in a significant number of recorded parking maneuvers, whereas the project primarily focuses on trajectory prediction during regular driving. Additionally, the roads in this scenario lack explicit lane markings, making it difficult to accurately identify driving lanes in both directions and associate vehicles with the correct map elements. Moreover, this scenario includes a high volume of pedestrians, with interactions between pedestrians and vehicles occurring primarily around zebra crossings—an area outside the scope of our focus.

Table 4-2 Features of the four scenarios [20]

Intersections	Features
Bendplatz: Figure 4-3 (a)	<ol style="list-style-type: none"> 1. Typical cross intersection with four arms and a priority road. 2. No regulated pedestrian crossings. 3. Common interaction between turning vehicles and VRUs like pedestrians and cyclists.
Frankenburg: Figure 4-3 (b)	<ol style="list-style-type: none"> 1. Cross intersection with four arms in a residential area and at a park. 2. With a zebra crossing and many parking lots. 3. No clear road centreline markings. 4. A high number of cyclists and pedestrians instead of vehicle trajectories. 5. More interactions between normal driving vehicles and vehicles moving in or out of parking spaces.
Heckstrasse: Figure 4-3 (c)	<ol style="list-style-type: none"> 1. T-junction in a suburb. 2. One main road with a left turn lane into the side road. 3. With cycle paths at the road edges and a pedestrian crossing in the middle. 4. Interactions due to vehicles making turns.
Neukoellner Strasse: Figure 4-3 (d)	<ol style="list-style-type: none"> 1. T-junction next to a bus depot in an industrial area. 2. Double-laned priority road in both directions with a left turn area into the side road. 3. Transit traffic due to proximity to a motorway access. 4. Providing more data regarding buses turning off. 5. Pedestrians going via a traffic island to one of the two bus stops.

Table 4-3 A summary of track numbers and components in four datasets

	numTracks	numVehicles	numVRUs
Bendplatz	3738	2503	1235
Frankenburg	6235	2436	3799
Heckstrasse	1279	1196	83
Neukoellner Strasse	2347	2098	249

All in all, the datasets of Bendplatz, Heckstrasse and Neukoellner Strasse are selected in our research because they provide sufficient trajectories of TAs and

variable manoeuvre patterns. They are denoted as Intersections A, B and C separately in the following sections.

The intersection of Frankenburg is neglected to avoid the complexity of a large number of interactions between VRUs and vehicles, given that the main focus of this project is the influence of the RL information.

4.3 Statistics Analysis of Selected Intersections

4.3.1 The Number of Road Users and Their Traversal Time

To have a general idea of the dataset, statistics of the data are given here. Table 4-4 shows the numbers of different types of road users in the target intersections. Parked vehicles were also logged in the dataset. In the analysis, more attention is attached to the moving objects, so these static road users have been excluded from the statistics. This explains the difference of numbers with Table 4-3.

Data imbalance is explicit considering the traffic agent types. It is clear that the dataset is dominated by the cars in all target intersections. The number of trucks is small in Intersections A and B while the bicyclists have fewer numbers in Intersections B and C. In Intersection A, the number of pedestrians is the largest among all three intersections.

Table 4-4 The track number of different actor classes in the target scenario

	Cars	Trucks	Bicyclists	Pedestrians
Intersection A	2383	58	434	794
Intersection B	1109	59	39	44
Intersection C	1854	221	86	163

Without traffic signals, these intersections lack strict vehicle ordering, leading to significant negotiation between vehicles at these locations. Intersection A potentially involves much more interaction between vehicles and VRUs, which will benefit the research to explore the influence of interaction on prediction accuracy. Most of the time, vehicles move within the driving lanes and try to obey

the traffic rules, which provides enough samples to study the influence of the RLs on trajectory prediction.

To have a better understanding of the traversal time of road users, the distributions of traversal time at intersections for each type of road user have been plotted, as shown in Figure 4-4.

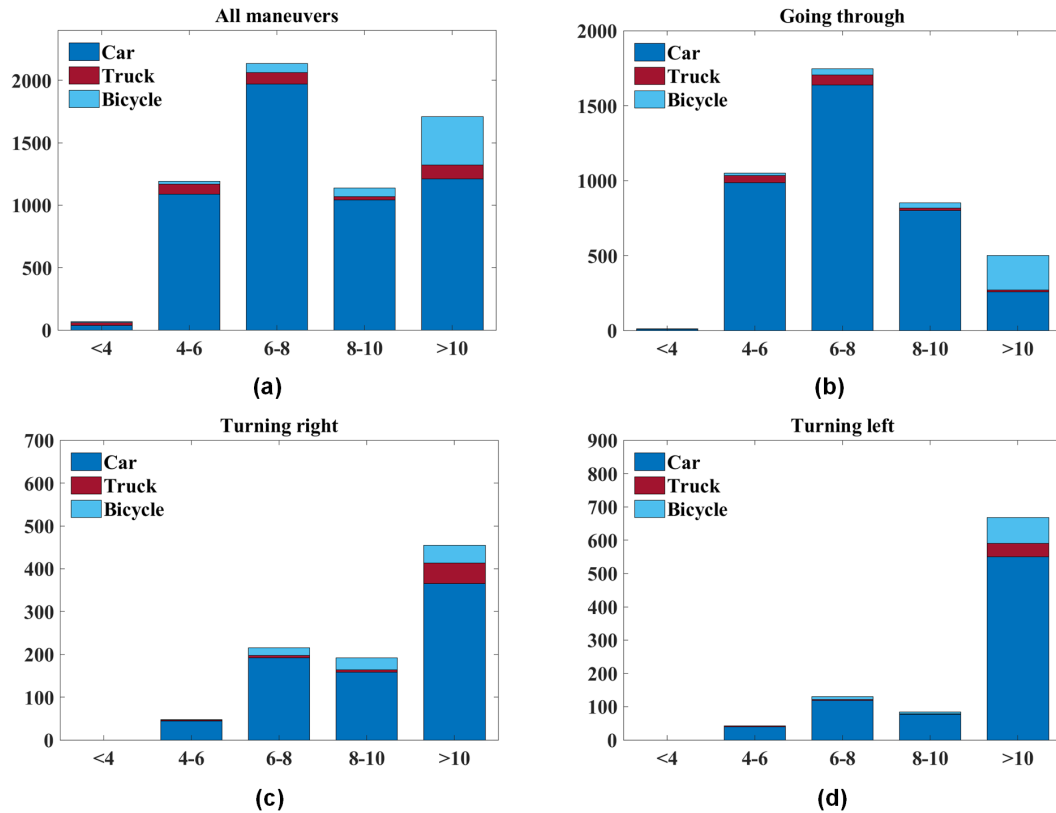


Figure 4-4 A summary of traffic agent data concerning traversal times

In the going-through scenario, road users have a shorter moving time compared to the other two manoeuvres. From the histogram, the cars, trucks and buses travel through the intersection below 8s. However, the bicyclists spend more time traversing the intersection because of their relatively low speed.

In the turning scenarios, both left and right ones, most road users have a large traversal time, even larger than 10s. This is because, in the unsignalized intersection, the turning manoeuvre should give priority to going through. In the

high traffic volume, road users tend to wait for a longer time to find a chance to finish left turning.

4.3.2 Analysis of Manoeuvres of Traffic Agents

To better illustrate the different manoeuvre types of traffic agents, and to have a better understanding of data in the inD, trajectories of different types of road users are clustered.

4.3.2.1 Classification of The Trajectories

By observation, road users have a limited selection of their entrances and exits to the intersection due to the geometric structure and traffic rules. In this case, trajectory clustering is operated according to both the initial position and final position of all actors. An example of possible entrance and exit lanes in Intersection A is illustrated in Figure 4-5.

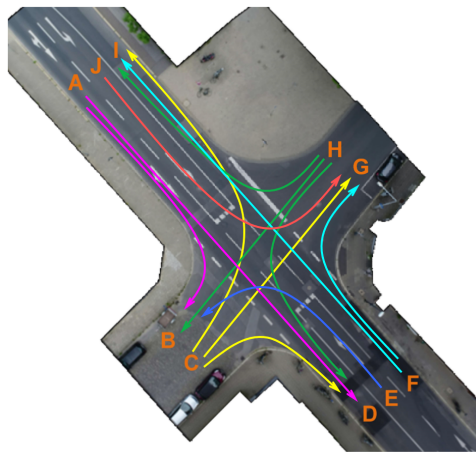


Figure 4-5 The potential entrance and exit for the road users. And the potential manoeuvres are marked.

The possible regions for the entrances and exits are selected manually according to the distribution of the data.

Trajectory clustering provides a foundational understanding of movement patterns. A summary of statistics of motion patterns in cars only is shown in Table 4-5.

Combining the statistics, a conclusion could be made that the driving-through pattern dominates the data (around 69.39%) and the remaining data are almost equally split into right- and left-turning patterns, with 15.18% and 15.43% separately.

Table 4-5 The summary of the number of tracks of different manoeuvres

	Car	Trucks	Bicycle	Overall
Right turning	759	62	89	910
Going through	3691	153	316	4160
Left turning	787	47	91	925

The results of trajectory clustering are shown in Figure 4-6, Figure 4-7 and Figure 4-8. Trajectories of cars, trucks and bicycles are plotted in blue, red and green separately.

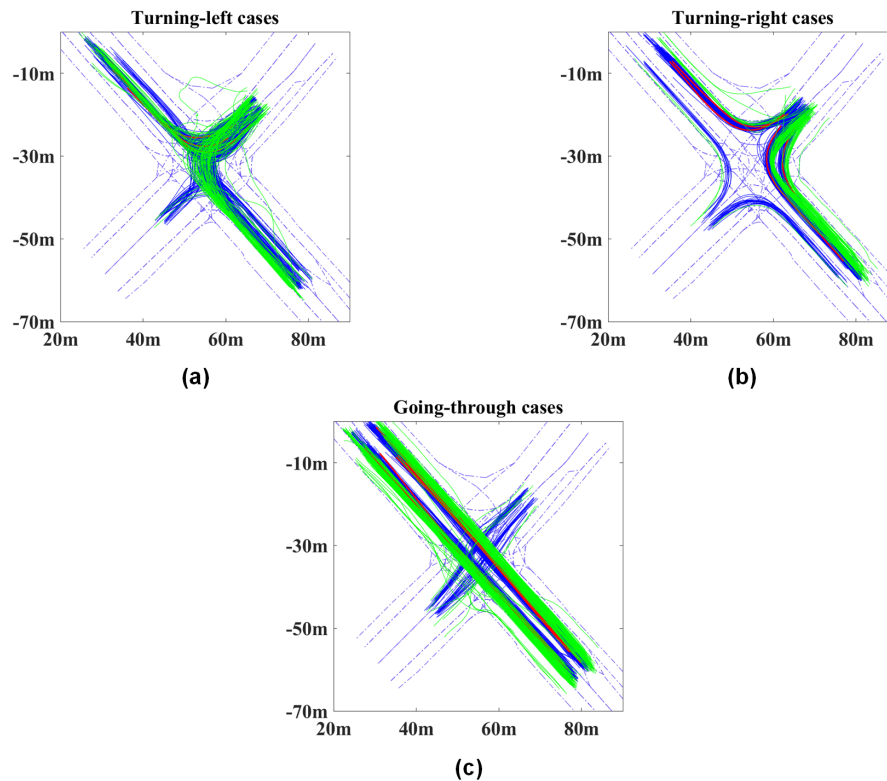


Figure 4-6 The clustering results at Intersection A.

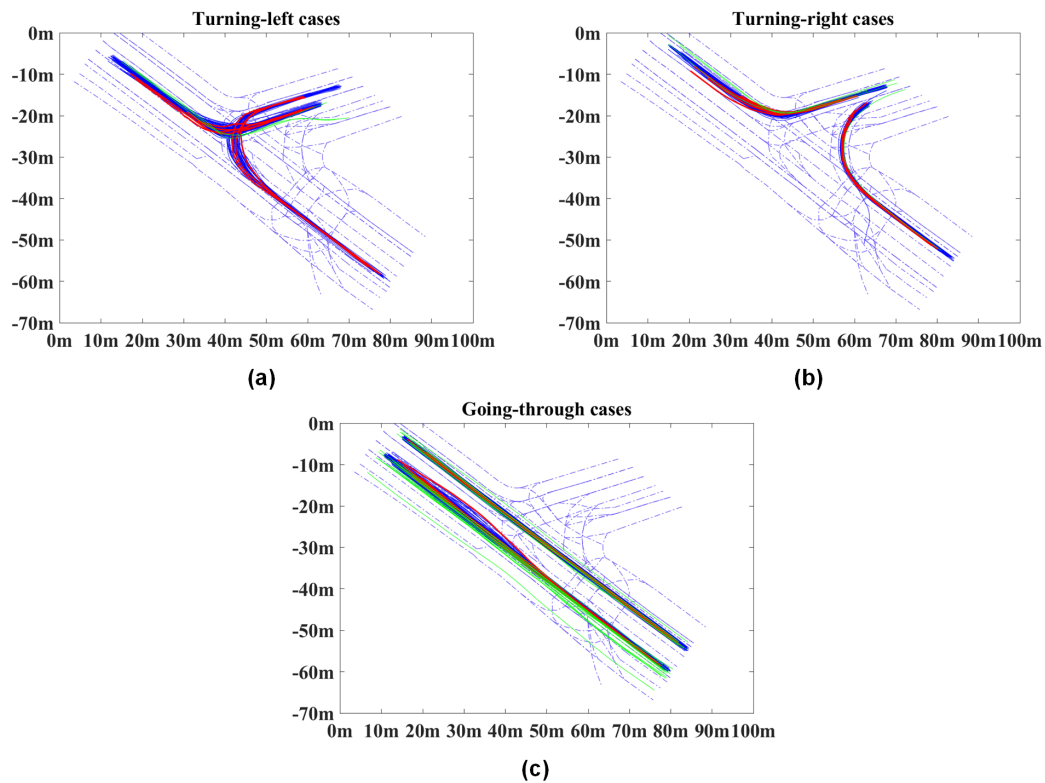


Figure 4-7 The clustering results at Intersection B.

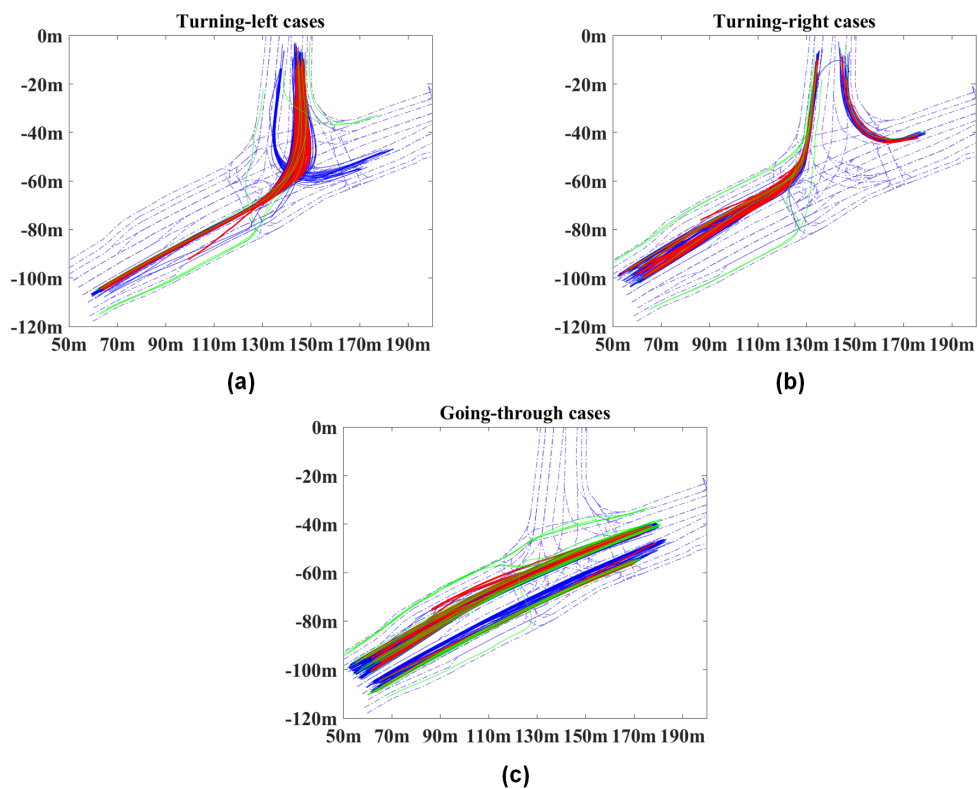


Figure 4-8 The clustering results at Intersection C.

4.4 Data Pre-processing

The inD dataset provides track data of all traffic agents in the recorded regions. However, these track data could not be used for DL-based model training without pre-processing. These recorded tracks feature varying lengths in the time domain, but the expected input to the trajectory prediction model is a fixed-length trajectory sequence. Thus, it is necessary to slice the original track data into tracklets of the same length, covering fixed observation and prediction horizons. Apart from slicing the tracks, some additional information should be appended to the track information to describe the traffic scenario around the chosen agent, including dynamic neighbours and static RL elements.

Although inD provides the RL maps of the target intersections, they are stored in a format that cannot be used directly in our selected research platforms. Additional works are required to parse the map files, extract map elements, rebuild the map topology and store the map data in a new structure which is easy and efficient to access and query. Also, map elements should be related to the trajectory points thus they could be used in further map-aware trajectory prediction.

Besides, track data and map element data should be standardised for the purpose of efficient and stable training. All details of the above-mentioned works are shown in the sections below.

4.4.1 Used tools

In this project, several software tools are used. An overview of used softwares and their related tasks is shown in Figure 4-9.

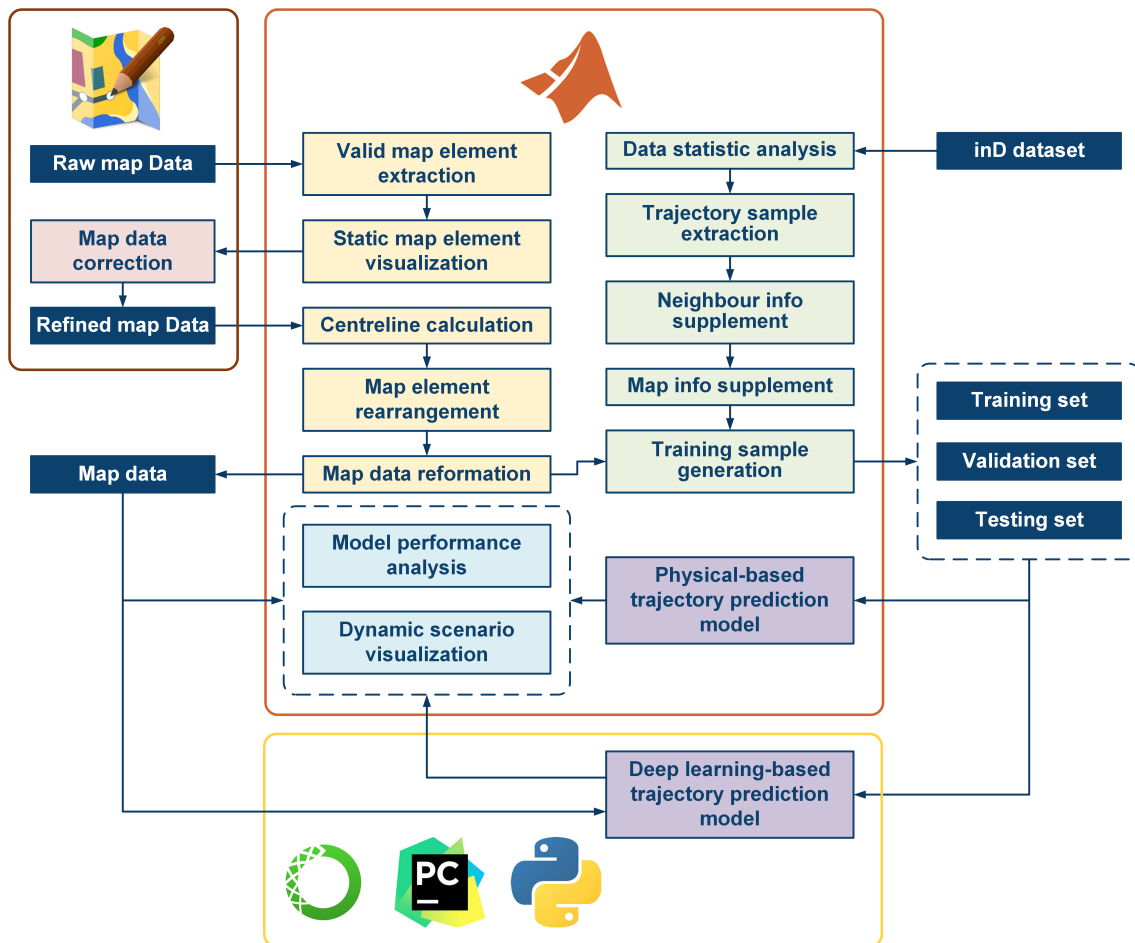


Figure 4-9 Used tools and related tasks

To accomplish the data pre-processing task, MATLAB is mainly used. The provided raw map data are parsed and analysed in MATLAB, where customised functions have been developed to extract valid map elements, calculate centrelines, rearrange map elements and reformat the map data. Additionally, JOSM is used to visualize both the original and refined map data with a real-world background. Detailed processing steps can be found in section 4.4.2.

Track data are also processed in MATLAB. Statistic analysis is performed to have an overview of inD dataset. Then, samples with fixed-length trajectories are produced, along with related traffic data like dynamic neighbours and static RL elements. Details can be found in section 4.4.3.

The processed map data and track data will be subsequently utilized in Python to train the DL-based trajectory prediction model. Besides, PyCharm serves as the primary Python development environment, and Anaconda is used to manage the Python environment effectively. On the contrary, trajectory prediction models based on physical principles are implemented in MATLAB.

4.4.2 Map Data Processing

4.4.2.1 Data Checking and Correction

In the inD dataset, the intersections feature complicated connections between arms. This complexity poses challenges in RL map creation, especially in the crossing areas where lanelets overlap and diverge. Given that Lanelet2 maps are manually created and maintained by providers, operational errors are inevitable.

Fundamental issues are incomplete definitions of lanelet, which will cause failures in lanelet identification during the map parsing process. For example, some lanelets lack one or two boundaries, as shown in Figure 4-10. This kind of error can only be fixed by referring to the original map data and adding appropriate boundary information.

```

<relation id='3010766' action='modify' visible='true' version='9'>
  <member type='way' ref='3010605' role='right' />
  <tag k='bicycle' v='yes' />
  <tag k='location' v='urban' />
  <tag k='one_way' v='no' />
  <tag k='region' v='de' />
  <tag k='subtype' v='walkway' />
  <tag k='type' v='lanelet' />
</relation>

```

→ Right boundary

(a)

```

<relation id='3010768' action='modify' visible='true' version='9'>
  <member type='way' ref='3010617' role='left' />
  <member type='way' ref='3010618' role='right' />
  <tag k='bicycle' v='yes' />
  <tag k='location' v='urban' />
  <tag k='one_way' v='no' />
  <tag k='region' v='de' />
  <tag k='subtype' v='walkway' />
  <tag k='type' v='lanelet' />
</relation>

```

→ Left and right boundaries

(b)

Figure 4-10 Typical definition of lanelets is listed. In (a), only the right boundary of the lanelet is assigned, leading to an incomplete definition of a lanelet. In (b), both left and right boundaries are given.

Another issue is that the points in the boundary are out of order, as shown in Figure 4-11. This error causes the break connection between the consecutive lanelets since they are connected according to the shared points and the point that should appear at the end is now in the middle of the sequence.

Other common errors include:

1. Duplication of lanelets: lanelets have the same geometry data like identical boundaries but are assigned with different IDs.
2. Mix-up between driveways and walkways: in the map, these two types of lanelets target vehicles and pedestrians separately. Incorrect classification will cause the connection between driveways and the walkways, which will cause misconstruction of driveways.
3. Misalignment of lanelet direction with lane direction. A complete road lane consists of several sequentially connected lanelets. These lanelets are connected by identifying shared points of their boundaries. For example,

two lanelets will be connected if the endpoint of its left boundary is identical to the start point of the other's left boundary. The misalignment of the lanelet left/right boundary and road lane left/right boundary will cause the misconnection of lanelets and further leads to the breakdown of lanes.

These errors are identified by the corruption in the automatic process of data extraction and can only be fixed manually.

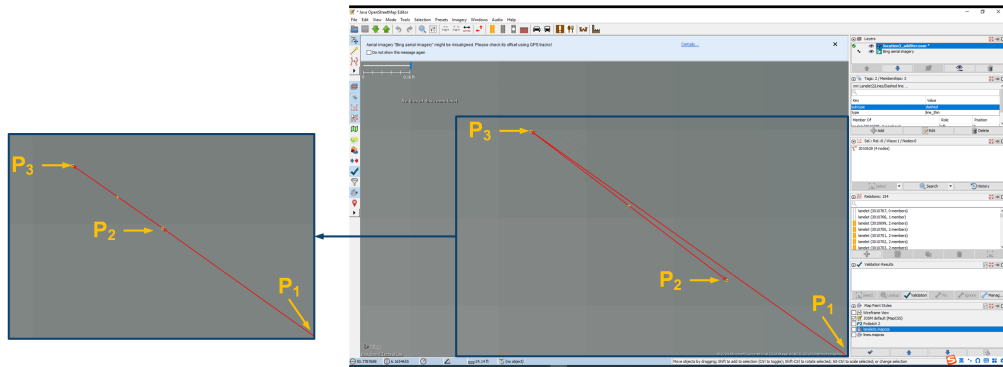


Figure 4-11 Disordered line points are shown on the right side, which forms a zigzag-like boundary. The order of these three points is then refined, which is shown on the left side.

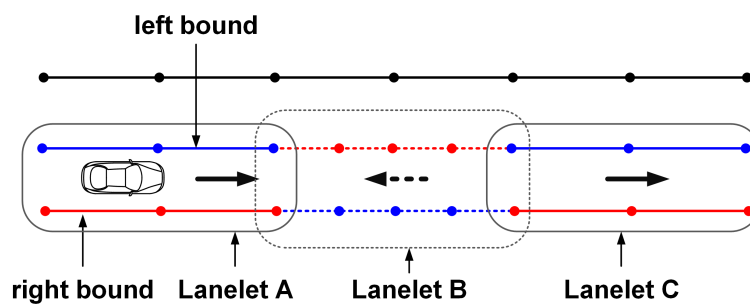


Figure 4-12 The directions of Lanelet A and C are aligned with that of the lane while Lanelet B is oriented in the opposite direction. The lane will be broken in connection because no shared points can be found between the left/right boundaries of Lanelet B and those of either Lanelet A or Lanelet C.

4.4.2.2 Map Data Extraction and Reformatting

The inD dataset provides RL maps of the target scenarios in the Lanelet2 format. To apply these map data in the following sections, there are some issues to be solved.

First, map elements like lanelets and multipolygons should be extracted individually from the original map files. In this research, MATLAB is chosen to do data pre-processing and Lanelet2 map files cannot be parsed correctly by MATLAB. MATLAB can import OpenStreetMap files and convert them into OpenDrive format for scenario reconstruction in the module of Driving Scenario Designer (DSD). Lanelet2 map element data are arranged similarly to OpenStreetMap data, both in file format and data structure. However, roads in OpenStreetMap and Lanelet2 are defined significantly differently. In OpenStreetMap, roads are defined by their centrelines while they are represented by road boundaries in the Lanelet2, as mentioned in Section 2.2. Forced loading will mislead MATLAB identifying road boundaries as road centrelines which causes distortion of original topologies and road geometries.

Second, the coordinates of the map data should be modified and aligned with those of the recorded track data. In the given Lanelet2 maps, WGS84 coordinates are used while the track data of traffic agents are given in UTM coordinates. The track data are offset according to a local origin, instead of the origin of the UTM zone. In the following sections, map data will be matched to tracks to relate these two kinds of information together. So, transfer should be done from WGS84 to a UTM origin.

Third, complete road lanes should be extracted from the Lanelet2 maps. Roads are separated into lane pieces which are represented by individual lanelets in Lanelet2 maps. Though complete roads could be found implicitly according to the connection between lanelets, they are not stored explicitly in the map but are important for centreline calculation in Chapter 5, in which centreline calculation is operated based on long roads instead of separate lanelets.

To remedy the above-mentioned issues, several steps are taken to process the data. The original Lanelet2 map files are parsed in MATLAB to extract lanelets and multi-polygons, which are basic map elements. Then corresponding boundaries and boundary points are related to these map elements. Additional tags to distinguish driving lanes and walkways are supplemented to each map element, according to relevant semantic tags in the original Lanelet2 files. At last, separate complete road lanes are extracted and the left and right neighbours of each lanelet are identified. The pipeline of general map data processing is shown in Figure 4-13, as well as the basic data structure to store some of the fundamental data.

In the data structure, the scenario IDs are used to distinguish the source of the map elements. This information is useful because map elements from multiple scenarios are stored in one file, which simplifies the data loading without need to load multiple files to import all scenarios. The driving lanelets are distinguished in the dataset by setting `Flag_drivingLane` to zero or one, representing walkways and driving lanes respectively. The parents and children represent the predecessors and successors of the lanelet, which are obtained during the construction of individual lanes from separate lanelets. The lateral adjacent neighbours (left and right) are identified by the shared boundaries of two lanelets. A target lanelet's left neighbour is identified if another lanelet has the same boundary as the target lanelet's left. And its right neighbour is identified similarly.

The complete lane is represented by sequential lanelets and its boundaries are concatenation of boundaries of sequential lanelets. The centreline calculation will be introduced in Chapter 5.

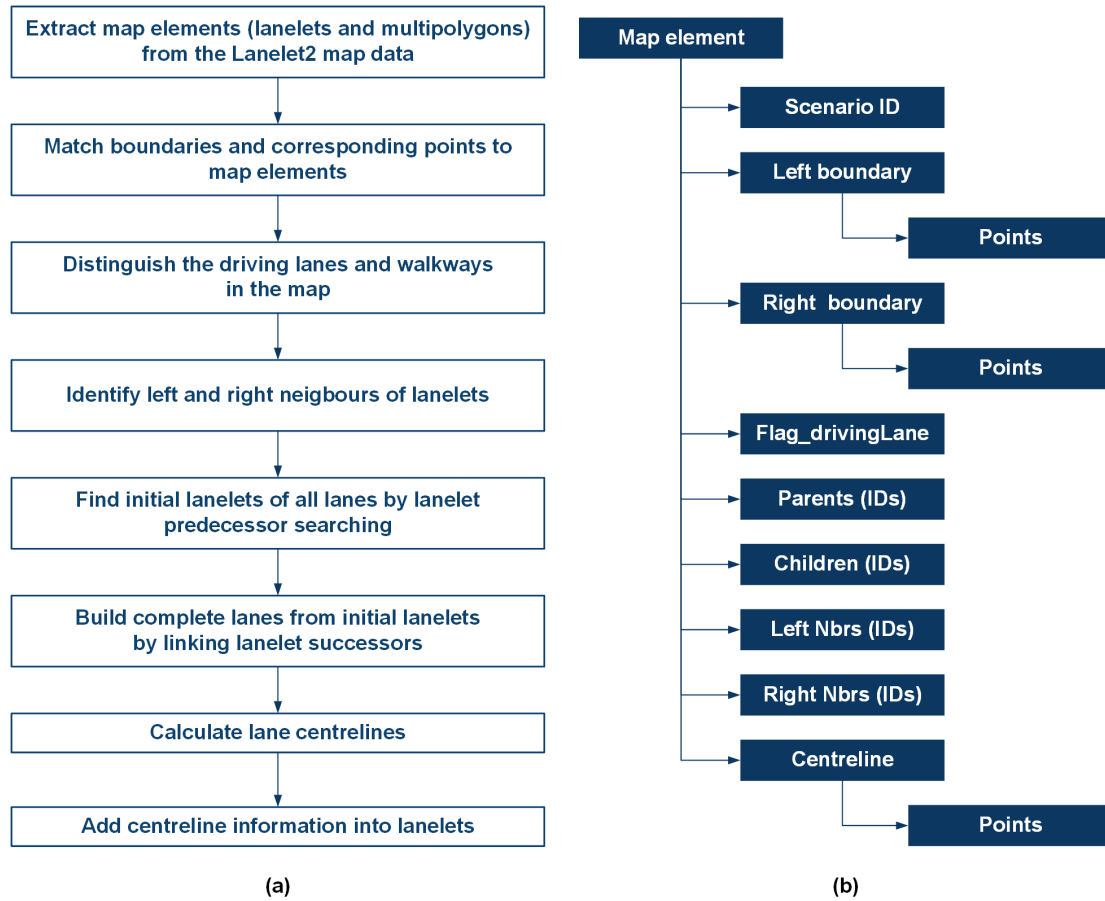


Figure 4-13 Pipeline of map data processing (a) and the customised data structure to store the map information (b).

4.4.3 Track Data Processing

Data fields contained in the original dataset are shown in Table 4-6. For this research, fields of dataset ID (DsID), actor ID (AcID), time frame (timeFr), x and y positions (P_x and P_y), and velocities in x and y directions (v_x and v_y) are retained. Scenario (Intersection) ID is also added to distinguish the traffic agents from different intersections. These heterogeneous data are stored in a row, representing a track point with semantic features of a TA at a specific time frame, as shown in Table 4-6. More information like neighbour information and map element information is appended to each row to describe the instant traffic scenario, reducing the number of searches across the database while this information is needed.

The whole lifecycle track of a traffic agent can be identified by its unique identifications, a combination of unique DsID and AcID, and occupies several rows in the sequential table, the number of which varies from agent to agent for different temporal lengths.

Table 4-6 An example of track data arranged in row

RowIdx	DsID	AcID	timeFr	ScelID	P_x	P_y	v_x	v_y	...
1	1	1	1	1					
2	1	1	2	1					
3	1	1	3	1					
...									
667	15	3	754	3					
668	15	3	755	3					
...									

4.4.3.1 Tracklet Extraction

This project aims to achieve long-term trajectory prediction with a typical prediction horizon exceeding 3 seconds as shown in Table 4-7. This prediction is based on a brief historical observation, the time of which varies depending on the scenarios. In highway applications, an observation horizon of 3 seconds is typical—longer than the 2-second horizon common in urban scenarios. This difference arises due to shorter observation times in urban areas, especially at intersections where traffic agents exhibit shorter track lifetimes. For urban scenarios, past observations of 2 seconds are often sufficient to capture essential motion context [64] and prediction models based on nuScenen [10] or Argoverse [8] commonly adopt this 2-second observation horizon.

A summary of commonly used observation and prediction horizons is listed below.

Table 4-7 A summary of observation and prediction horizons

Literature	Observ. Horizon	Pred. Horizon	Sce.	Dataset
[67] [83] [85] [86] [104] [106] [101] [109] [107] [111] [112]	2	3	Urban	Argoverse
[54]	2	3	Urban	nuScenes
[64]	2	4	Urban	KITTI
[66] [97] [98] [108] [113]	2	6	Urban	nuScenes
[62]	3.2	4.8	Mixed	inD, highD, rounD
[58]	0.56	4.8	Roundabout	-
[73]	2	4	Highway	NGSIM
[49]	3	1	Highway	NGSIM
[72]	3	2	Highway	NGSIM
[59]	3	2	Highway	-
[60] [70] [71] [87] [1] [34] [92]	3	5	Highway	NGSIM
[57]	5	5	Highway	NGSIM

In this project, we adopt a 2-second historical observation horizon and a 4-second prediction horizon, as also used in [64]. Given that most recorded moving objects have a travel time of around 6 seconds as shown in Figure 4-4, setting a prediction horizon beyond 4 seconds would require recorded samples with trajectories longer than 6 seconds to ensure fully complete trajectory segments for both observation and prediction periods. And this choice will cause insufficient

extracted training samples from the dataset. Thus, considering these factors, this project sets the historical observation and prediction horizons at 2 seconds and 4 seconds, respectively.

This fixed-duration tracklet, 6s in total, is an individual sample in the model training process.

A complete track of a traffic agent is captured immediately from its entrance to the recorded region and it disappears when it moves out of the perception field of the drone camera. From agent to agent, they have diverse mobilities and perform distinct manoeuvres. As a result, agents in the inD have varying temporal lengths.

Based on these individual tracks, a sliding window covering track data of 6s is applied to extract sample tracklets for model training purposes. This sliding window starts from the start of the track until it reaches the track end, by which way multiple training samples are extracted from a complete agent track. An illustration of this process is shown in Figure 4-14.

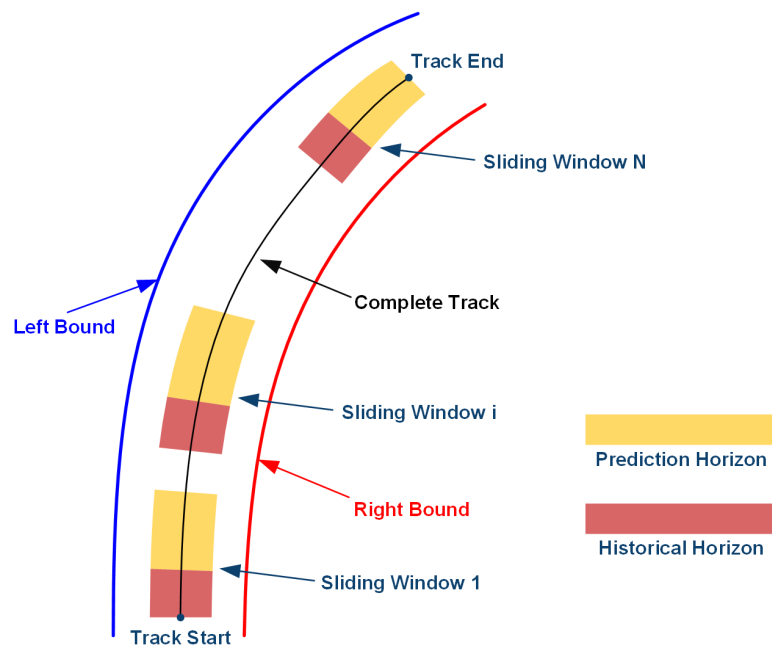


Figure 4-14 An illustration of sample extraction through sliding windows

In the inD dataset, data are recorded at 25 Hz. To lower this sampling rate, a skip factor of 5 is deployed at the raw data, leading to a new sampling rate of 5 Hz. With a historical observation of 2s and a prediction horizon of 4s, 11 and 20 track points are contained in the separate track pieces separately.

4.4.4 Data Supplement

In addition to the agent track data, there is other information that should be supplemented to describe both the dynamic and static traffic scenes, like the information regarding SAs and RL elements.

4.4.4.1 Neighbour Identification

Neighbours of TA are identified according to the method used in [1]. SAs are identified if they are located in a grid centred at the TA. This grid is 13 by 3 and each cell in the grid has a size of 5 m by 3.5 m, as shown in Figure 4-15.

If no other agent is identified in one of the grids, zeros will be supplemented as the placeholder.

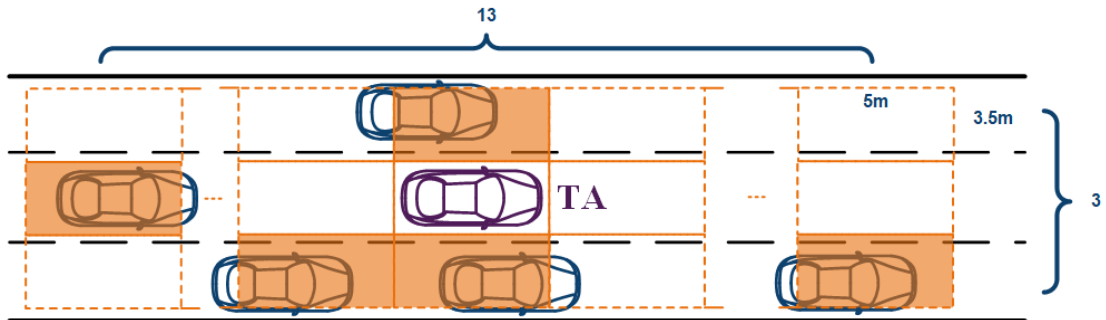


Figure 4-15 An example of the 13by3 grid. The filled polygons stand that a SA has been identified in grid cells.

4.4.4.2 Match The Position Data to The Map Elements

The position of the trajectory point will be matched with map elements in the Lanelet2 map. The strategy is to identify whether a track point is inside the polygon constrained by the map element boundaries. If the track point is included

by several overlapping map elements, all map elements will be related to this track point, as shown in Figure 4-16.

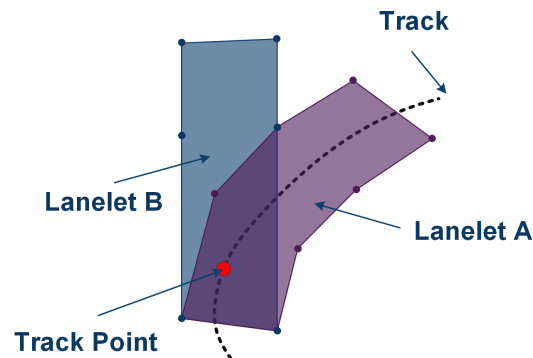


Figure 4-16 The track point (red dot) will be matched to both Lanelet A and B because it is contained within both of them. This overlapping of map elements usually happens at merging or splitting lanes and at intersections.

4.4.5 Dataset Splitting for DL Purpose

Agent data are split into training, validation and testing sets. To minimise the imbalance considering 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 has been used in [129] [130]. Detailed statistics of samples are shown in Table 4-8.

Table 4-8 The statistics of actor samples in each scenario

Actor Class	Mnvr	Intersection A			Intersection B			Intersection C		
		Trng	Valid	Tstg	Trng	Valid	Tstg	Trng	Valid	Tstg
Cars	GTH	692	185	186	564	151	146	1373	378	366
	TL	463	126	129	108	30	29	169	44	48
	TR	386	106	101	101	24	30	222	60	61
Trucks	GTH	27	6	8	12	4	4	57	15	15
	TL	3	0	1	7	2	2	35	8	12
	TR	4	0	1	8	3	2	48	14	12
Bicycles	GTH	250	64	74	21	7	6	29	7	7

TL	81	20	25	3	2	1	7	1	3
TR	74	19	22	4	1	1	8	2	2
Overall	1980	526	547	828	224	221	1948	529	526

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

4.4.6 Standardisation

The standardised variables include position data (P_x and P_y), and velocity data (v_x and v_y). The standardisation process is as follows:

$$z = \frac{x - \mu}{\sigma} \quad (4-1)$$

where μ is the mean value of the input feature x , σ is the standard deviation of the feature values, z is the final standardised feature. The statistical mean and standard deviation of each variable are calculated across the training sample.

This transformation is also called Z-score normalisation to have a mean of zero and a standard deviation of one.

Besides, RL map element features are also standardised. The positions of sequential centre points of Lanelets are used to represent the map element feature. The coordinates of these centre points are standardised by reusing the standardisation parameters of x and y positions, given that the track position coordinates and Lanelet centre point coordinates in one selected intersection have similar value ranges. This method helps to keep the statistic characters aligned between track data and map data.

4.5 Trajectory Prediction Model Design Based on Kinematic Models

4.5.1 Introduction of Physical-based Models

As described in section 3.2, given the current state of the TA and its transition process, a trajectory prediction model can be established. This transition process is ruled by physics laws, described by kinematic or dynamic models.

4.5.1.1 Dynamic Models

Dynamic models, based on Lagrange's equations, describe motion by considering various forces affecting an agent, such as longitudinal and lateral tyre forces. A detailed dynamic model includes tyre characteristics and internal parameters like suspension settings and steering system structures, making them suitable for control-oriented applications. However, for trajectory prediction, more efficient models are preferred. These models often use a "bicycle" representation, which simplifies a vehicle to a two-wheeled front-wheel-drive vehicle on a 2-D plane [32]. Examples of these simplified dynamic models can be found in [131].

The internal parameters of a vehicle required by dynamic models are not observable by exteroceptive sensors, and the control inputs of target vehicles are usually unknown. This limitation excludes the use of dynamic models for ITS applications involving vehicles other than the ego vehicle.

4.5.1.2 Kinematic Models

Kinematic models describe a vehicle's motion using mathematical relationships between parameters such as position, velocity, and acceleration, without considering the underlying forces. They neglect friction forces and assume that each wheel's velocity is aligned with its direction [132]. Due to their simplicity and sufficient accuracy for non-control applications like trajectory prediction, kinematic models are more popular than dynamic models. A comprehensive survey of these kinematic models was conducted by Schubert et al [133].

The simplest kinematic models include the Constant Velocity (CV) [29] and Constant Acceleration (CA) [134] models, both of which assume straight-line

motion for agents. More advanced models, such as the Constant Velocity and Turn Rate (CVTR) and Constant Acceleration and Turn Rate (CATR) models, account for variations around the z-axis by incorporating yaw angle and yaw rate variables into the agent's state vector [28]. Despite this added complexity, these models maintain low computational demands by keeping velocity and yaw rate decoupled.

Using kinematic models for agent state evolution, a straightforward approach to trajectory prediction is to apply these models to the agent's current state, assuming perfect knowledge of this state. This single-forward simulation method is computationally efficient, making it suitable for applications with strict real-time constraints. However, it does not account for uncertainties in the current state or the limitations of the evolution model [32]. Consequently, the predicted trajectories are unreliable for long-term predictions (beyond 1s).

4.5.1.3 Shortcomings

Physics-based motion models, which rely on dynamic and kinematic properties, are limited to short-term motion prediction (less than 1 second). They typically cannot anticipate changes in agent motion due to specific manoeuvres (e.g., slowing down, accelerating to make a turn) or external factors (e.g., slowing down due to a frontside agent). The necessary inputs for these prediction models are often unknown, like the control inputs for dynamic models. While physics-based methods are computationally efficient, they fail to account for complex scene factors like interactions between agents, resulting in poor accuracy. Consequently, these methods are best suited for simple scenarios with short-term prediction horizons (less than 1 second).

4.5.2 Dealing with Uncertainties by Kalman Filters

Uncertainty in the current vehicle state and its evolution can be modelled using a normal distribution [29]. Under this distribution, Kalman filters are commonly used to deal with these uncertainties. The Kalman filter is a mathematical algorithm that provides an efficient way to estimate the state of a system from a series of

noisy measurements. It is particularly effective for systems with Gaussian noise, and it provides not only the state estimate but also the uncertainty of the estimate, which is useful for assessing the reliability of predictions in real-time applications. It operates in two main steps: prediction and correction.

1. **Prediction:** The filter predicts the state of the system at the next time step based on the current state and the system's dynamic model. This prediction includes both the system's state (e.g., position, velocity) and its associated uncertainty, which is represented by a covariance matrix.
2. **Correction:** When new measurements are available, the filter updates its prediction by blending the predicted state with the actual measurement. The combination is weighted according to the uncertainty of both the prediction and the measurement. This step ensures that the estimate is refined, reducing errors introduced by noisy or incomplete data.

In the context of trajectory prediction, the Kalman filter is used to improve the accuracy of estimating the current state of a vehicle based on its historical states (e.g., past positions and velocities). For example, if a vehicle's position and velocity are measured with noise, the Kalman filter can estimate a more accurate current state by considering the historical data and the model's dynamics. This refined state is then used to predict the vehicle's future trajectory under the assumption that the velocity or turning rate remains constant [29] [135] [136].

For linear systems, the Kalman filter performs well. However, many real-world applications involve non-linear dynamics, such as vehicle motion with turning. In such cases, the Extended Kalman Filter (EKF) is employed. The EKF adapts the Kalman filter to non-linear systems by approximating non-linear functions with their first-order Taylor expansion, using the Jacobian matrix to linearise the system around the current estimate. This enables the filter to handle non-linear dynamics by making small, linear adjustments to the state estimation.

4.5.3 Problem Statement

The problem of trajectory prediction based on the kinematic models is to anticipate the traffic agents' states, which in our case are positions, headings and

relevant velocities and turning rates, in the future time steps on the assumption that the agents' movement (like the velocities and turning rate) in the prediction horizon will keep unchanged as the current.

Considering that agents' states are obtained by measurements from relevant sensors, the measured states come with sensor noises, which induces uncertainties in the prediction process. To lower the influence of measurement uncertainties, the Kalman Filter is used to do a better estimation of the current states S_t , from a list of historical observations $z_{t-h}, z_{t-h+1}, \dots, z_t$ associated with the same object. Then, this estimated state $\hat{S}_t^+$ is used in the future prediction task.

In our case, this is the whole pipeline combining the above two stages:

1. Extract measurements in the past 2s (the historical horizon) and feed them into the Kalman filter to do an estimation of the current state $\hat{S}_t^+$.
2. Apply the estimated state $\hat{S}_t^+$ into the kinematic model to predict future states of the traffic agent.

In this method, the future motion of the target agent (TA) is predicted by linking control inputs (e.g. steering, acceleration), agent properties (e.g. weight), statuses (e.g. position, speed, heading), and external conditions (e.g. friction coefficient of the road surface) to the evolution of the statuses of the agent (e.g. position, speed, heading) [32]. The complexity of the models varies based on the granularity of the representation of agent dynamics and kinematics, as well as how state transitions are handled.

4.5.4 Applied Kinematic Models

This research employs two commonly used kinematic models. The first model assumes that velocities remain constant, known as the constant velocity model (CV). The second model assumes that both velocities and turning rates remain constant, referred to as the constant velocity and turning rate model (CVTR). Both models are used consistently for state estimation and trajectory prediction.

4.5.4.1 The Constant Velocity Model

In the CV model, the velocities of the target agent in the x and y directions, named v_x and v_y separately, remain the same in the process of state transition. An illustration of the transition of traffic agent's states during time $t - 1$ and t is shown in Figure 4-17.

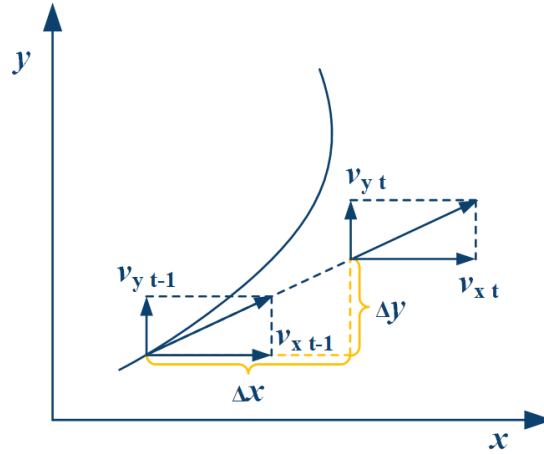


Figure 4-17 The illustration of the state transition in the CV model at time $t-1$ and t

The increments in the x and y directions could be calculated as below:

$$\Delta x = v_x \cdot d_T \quad (4-2)$$

$$\Delta y = v_y \cdot d_T \quad (4-3)$$

4.5.4.2 The Constant Velocity and Turning Rate Model

This constant model is based on the assumption that the longitudinal velocity v_{lon} of the target agent between time $t - 1$ and t maintain the same, and the traffic agent changes its heading angle θ at a constant angular velocity ω_{TR} around a fixed point. An illustration of the transition of traffic agent's states during time $t - 1$ and t is shown in Figure 4-18. Due to the constant longitudinal velocity and turning rate, the road user's locus is along an arc, whose radius is R .

$$R = \frac{v_{lon}}{\omega_{TR}} \quad (4-4)$$

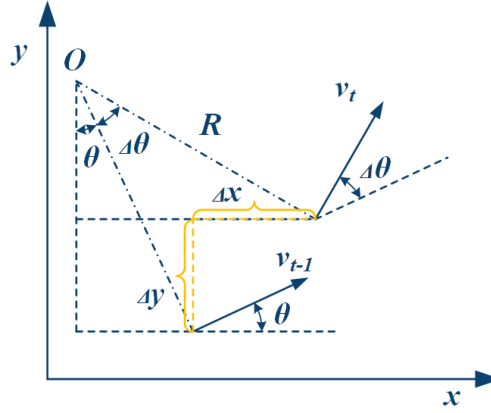


Figure 4-18 The illustration of the state transition in the CVTR model at time t-1 and t

The increments in the x and y directions, as well as in the heading angle θ , could be calculated as below:

$$\Delta\theta = \omega_{TR} \cdot d_T \quad (4-5)$$

$$\Delta x = R \sin(\theta + \Delta\theta) - R \sin(\theta) \quad (4-6)$$

$$\Delta y = R \cos(\theta) - R \cos(\theta + \Delta\theta) \quad (4-7)$$

4.5.5 Deployment of Kalman Filters with The Kinematic Models

4.5.5.1 Kalman Filter with The Constant Velocity Model

The state S contains the x and y coordinates, the velocity v_x in the x direction and the velocity v_y in the y direction. The state matrix S_t is listed below:

$$S_t = \begin{bmatrix} x_t \\ y_t \\ v_{x\ t} \\ v_{y\ t} \end{bmatrix} \quad (4-8)$$

The differential equations are listed in the form of matrixes below:

$$S_t = \Phi S_{t-1} + w_{t-1} \quad (4-9)$$

$$\begin{bmatrix} x_t \\ y_t \\ v_{x\ t} \\ v_{y\ t} \end{bmatrix} = \begin{bmatrix} x_{t-1} + v_{x\ t-1} \cdot d_T \\ y_{t-1} + v_{y\ t-1} \cdot d_T \\ v_{x\ t-1} \\ v_{y\ t-1} \end{bmatrix} \quad (4-10)$$

where S_t is the state vector in the time frame of the subscript, Φ is the state-transition matrix and w_{t-1} is the process noise in the time frame of the subscripts.

The state-transition matrix is listed below,

$$\Phi = \begin{bmatrix} 1 & 0 & d_T & 0 \\ 0 & 1 & 0 & d_T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4-11)$$

where d_T is the constant time interval.

Considering the velocities in the inD dataset are deduced from position measurement data, the measurement model is set to be:

$$z_t = HS_t + v_t \quad (4-12)$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (4-13)$$

where z_t is the measurement at time t , H is the measurement matrix, S_t is the state vector at time t and v_t is the measurement noise vector.

The prediction functions are:

$$\hat{S}_t^- = \Phi \hat{S}_{t-1}^+ \quad (4-14)$$

$$P_t^- = \Phi P_{t-1} \Phi^T + Q \quad (4-15)$$

where $\hat{S}_t^-$ is the predicted state at time t , $\hat{S}_{t-1}^+$ is the updated state estimation at time $t - 1$, P_t^- is the predicted state estimation error covariance matrix at time t , Q is the process noise covariance matrix and P_{t-1} is the state estimation error covariance matrix.

The update functions are:

$$K_t = \frac{P_t^- H^T}{H P_t^- H^T + R} \quad (4-16)$$

$$\hat{S}_t^+ = \hat{S}_t^- + K_t(z_t - H\hat{S}_t^-) \quad (4-17)$$

$$P_t = (I - K_t H) P_t^- \quad (4-18)$$

where K_t is the Kalman gain, R is the measurement noise covariance matrix, $\hat{S}_t^+$ is the updated state estimation, $\hat{S}_t^-$ is the predicted state estimation, P_t is the updated state estimation error covariance matrix and I is the identity matrix.

4.5.5.2 Extended Kalman Filter with The Constant Velocity And Turning Rate Model

In this case, the state S contains the x and y coordinates, the heading angle θ , the longitudinal velocity v_{lon} and the angular velocity ω_{TR} . The state matrix S_t is listed below:

$$S_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ v_{lon\ t} \\ \omega_{TR\ t} \end{bmatrix} \quad (4-19)$$

According to the equations of Δx and Δy in (4-6) and (4-7), the transition equations are nonlinear. Thus, the extended Kalman Filter (EKF) is selected to estimate the states of road users.

The discrete dynamics of the system are listed below:

$$S_t = f(S_t, d_T) \quad (4-20)$$

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \\ v_{lon\ t} \\ \omega_{TR\ t} \end{bmatrix} = f(S_t, d_T)$$

$$= \begin{bmatrix} x_{t-1} + \frac{v_{lon\ t-1}}{\omega_{TR\ t-1}} \sin(\theta_{t-1} + \omega_{TR\ t-1} \cdot d_T) - \frac{v_{lon\ t-1}}{\omega_{TR\ t-1}} \sin(\theta_{t-1}) \\ y_{t-1} + \frac{v_{lon\ t-1}}{\omega_{TR\ t-1}} \cos(\theta_{t-1}) - \frac{v_{lon\ t-1}}{\omega_{TR\ t-1}} \cos(\theta_{t-1} + \omega_{TR\ t-1} \cdot d_T) \\ \theta_{t-1} + \omega_{TR\ t-1} \cdot d_T \\ v_{lon\ t-1} \\ \omega_{TR\ t-1} \end{bmatrix} \quad (4-21)$$

The Jacobian of the state-transition model is:

$$F = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} & \frac{\partial f}{\partial v_{lon}} & \frac{\partial f}{\partial \omega_{TR}} \end{bmatrix} \quad ($$

4

-

2

2

)

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad ($$

4

-

2

3

)

$$\frac{\partial f}{\partial y} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad ($$

4

-

2

4

)

$$\frac{\partial f}{\partial \theta} = \begin{bmatrix} \frac{v_{lon\ t-1}}{\omega_{TR\ t-1}} [\cos(\theta_{t-1} + \omega_{TR\ t-1} \cdot d_T) - \cos(\theta_{t-1})] \\ \frac{v_{lon\ t-1}}{\omega_{TR\ t-1}} [-\sin(\theta_{t-1}) + \sin(\theta_{t-1} + \omega_{TR\ t-1} \cdot d_T)] \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{pmatrix} (\\ 4 \\ - \\ 2 \\ 5 \\) \end{pmatrix}$$

$$\frac{\partial f}{\partial v_{lon}} = \begin{bmatrix} \frac{\sin(\theta_{t-1} + \omega_{TR\ t-1} \cdot d_T) - \sin(\theta_{t-1})}{\omega_{TR\ t-1}} \\ \frac{\cos(\theta_{t-1}) - \cos(\theta_{t-1} + \omega_{TR\ t-1} \cdot d_T)}{\omega_{TR\ t-1}} \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (4-26)$$

$$\frac{\partial f}{\partial \omega} = \begin{bmatrix} -\frac{v_{lon\ t-1}}{(\omega_{TR\ t-1})^2} \sin(\theta_t) + \frac{v_{lon\ t-1}}{\omega_{TR\ t-1}} \cos(\theta_t) d_T + \frac{v_{lon\ t-1}}{(\omega_{TR\ t-1})^2} \sin(\theta_{t-1}) \\ -\frac{v_{lon\ t-1}}{(\omega_{TR\ t-1})^2} \cos(\theta_{t-1}) + \frac{v_{lon\ t-1}}{(\omega_{TR\ t-1})^2} \cos(\theta_t) + \frac{v_{lon\ t-1}}{\omega_{TR\ t-1}} \sin(\theta_t) d_T \\ d_T \\ 0 \\ 1 \end{bmatrix} \quad (4-27)$$

The measurement model is:

$$z_t = HS_t + v_t \quad (4-28)$$

where z_t is the measurement at time t , H is the measurement matrix, S_t is the state vector at time t and v_t is the measurement noise vector.

Since the angular velocity data are not provided by the original dataset, it is excluded from the observation matrix. And the velocity is also excluded for it is derived from the position data. The Jacobian of the measurement model is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (4-29)$$

The prediction functions are:

$$\hat{S}_t^- = f(\hat{S}_{t-1}^+) \quad (4-30)$$

$$P_t^- = F P_{t-1} F^T + Q \quad (4-31)$$

where $\hat{S}_t^-$ is the predicted state at time t , $\hat{S}_{t-1}^+$ is the updated state estimation at time $t - 1$, P_t^- is the predicted state estimation error covariance matrix at time t , Q is the process noise covariance matrix and P_{t-1} is the state estimation error covariance matrix.

The update functions are:

$$K_t = \frac{P_t^- H^T}{H P_t^- H^T + R} \quad (4-32)$$

$$\hat{S}_t^+ = \hat{S}_t^- + K_t (z_t - H \hat{S}_t^-) \quad (4-33)$$

$$P_t = (I - K_t H) P_t^- \quad (4-34)$$

where K_t is the Kalman gain, R is the measurement noise covariance matrix, $\hat{S}_t^+$ is the updated state estimation, $\hat{S}_t^-$ is the predicted state estimation, P_t is the updated state estimation error covariance matrix and I is the identity matrix.

4.5.6 Trajectory Prediction with The Kinematic Models

4.5.6.1 Constant Velocity Model

The updated states of the traffic agent at each prediction step are as follows:

$$\begin{bmatrix} x_{t+hrzn} \\ y_{t+hrzn} \\ v_{x\ t+hrzn} \\ v_{y\ t+hrzn} \end{bmatrix} = \begin{bmatrix} x_{t-1} + v_{x\ t} \cdot d_T \cdot N_{hrzn} \\ y_{t-1} + v_{y\ t} \cdot d_T \cdot N_{hrzn} \\ v_{x\ t} \\ v_{y\ t} \end{bmatrix} \quad (4-35)$$

where N_{hrzn} is the prediction step.

4.5.6.2 Constant Velocity And Turning Rate Model

According to the assumption, the longitudinal velocity v_{lon} and the angular velocity ω_{TR} will not change in the whole prediction horizon. Thus, the traffic agents' states at each prediction step are like this:

$$\begin{bmatrix} x_{t+hrzn} \\ y_{t+hrzn} \\ \theta_{t+hrzn} \\ v_{t+hrzn} \\ \omega_{t+hrzn} \end{bmatrix} = \begin{bmatrix} x_t + \frac{v_{lon} t}{\omega_t} \sin(\theta_t + \omega_t \cdot d_T \cdot N_{hrzn}) - \frac{v_{lon} t}{\omega_t} \sin(\theta_t) \\ y_t + \frac{v_{lon} t}{\omega_t} \cos(\theta_t) - \frac{v_{lon} t}{\omega_t} \cos(\theta_t + \omega_t \cdot d_T \cdot N_{hrzn}) \\ \theta_t + \omega_t \cdot d_t \cdot N_{hrzn} \\ v_t \\ \omega_t \end{bmatrix} \quad (4-36)$$

4.6 Trajectory Prediction Model Design Based on DL Models

The problem of trajectory prediction is formulated as determining the future trajectories of a target agent (TA) conditioned on its track history and useful scenario information (like neighbours and RL maps) at each discrete time step t . Given observations X_{status} in the form of past trajectories and scenarios of TAs, the prediction model seeks to provide their future trajectories. In this research, TAs include cars, buses and bicycles in the urban scenarios.

4.6.1 Inputs and Outputs

Formally, a set of observable features X_{status} and a set of target outputs 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. Let $T_{hist} = \{0, \dots, h\}$ and, for $k \in T_{hist}$, the value of the TA's status k time steps earlier is denoted by $X_{status}^{(t-k)}$. Similarly, define $T_{pred} = \{1, \dots, f\}$ and, for $k \in T_{pred}$, denote the value of output time steps in the future by $Y^{(t+k)}$. Uppercase X_{status} and Y are used to respectively denote the previously observed status features and corresponding predicted outputs. It is proposed to use a DL-based model, in which a regression function F is trained such that the predicted outputs $\hat{Y} = F(X_{status})$ match the actual values Y as closely as possible.

The inputs to the model are track histories of the target traffic agent:

$$X_{status} = [X_{status}^{(t-h)}, \dots, X_{status}^{(t-1)}, X_{status}^{(t)}], \quad (4-37)$$

where h is a fixed time horizon, and at any time instant t , for the TA, the following features are defined in $X_{status}^{(t)}$:

- Absolute position in the x and y direction with a local Cartesian coordinate: P_x and P_y ,
- Velocities in the x and y directions: v_x and v_y ,
- SAs' information SV ,
- Map information M

$$X_{status}^{(t)} = [P_x^{(t)}, P_y^{(t)}, v_x^{(t)}, v_y^{(t)}, SV^{(t)}, M^{(t)}]. \quad (4-38)$$

$$SV^{(t)} = [P_{x1}^{(t)}, P_{y1}^{(t)}, v_{x1}^{(t)}, v_{y1}^{(t)}, \dots, P_{xn}^{(t)}, P_{yn}^{(t)}, v_{xn}^{(t)}, v_{yn}^{(t)}], \quad (4-39)$$

where subscript n denotes the index of the SA.

In this research, map elements are represented by their centrelines. Thus, the map information M is an assembly of several centrelines in the target scenario, each consisting of multiple centrepointh containing coordinate information. For efficiency, K nearest centrelines are concluded, instead of the complete RL map. The distance between the TA and the candidate centreline is represented by the distance from the TA to the start point of that centreline, which indicates the More specifically, $K = 12$, which enables sufficient search in both lateral and longitudinal directions of the lane.

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

$$Y = [Y^{(t+1)}, \dots, Y^{(t+t_f)}], \quad (4-40)$$

and at each time frame t ,

$$Y^{(t)} = [P_x^{(t)}, P_y^{(t)}] \quad (4-41)$$

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

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 importance of this additional velocity input will be revealed in 6.3.

4.6.2 Useful methods

4.6.2.1 Long Short-Term Memory (LSTM)

LSTM is popular because of its superiority in alleviating gradient vanishing issues among other RNN methods [137]. This capability is realised by three gate operations: forget gate, input gate and output gate, as Figure 4-19 shows.

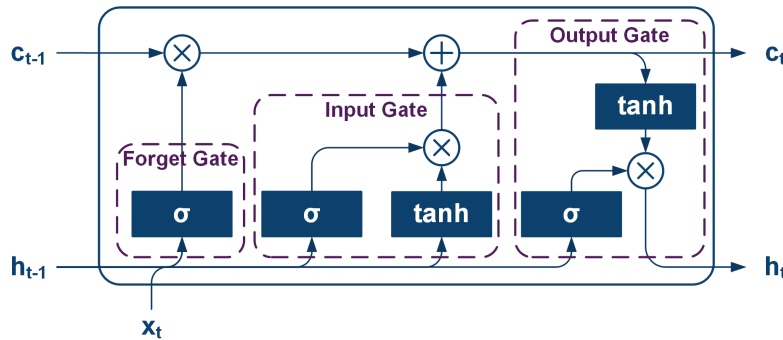


Figure 4-19 Schematic of LSTM. C_t is the cell state of the LSTM cell. x_t is the input at the current time frame. h_t is the hidden state. σ is the *sigmoid* activation function.

The information flow in one LSTM cell relies on these three gates and the cell state C_{t-1} , which is a memory that flows through previous LSTM cells. In detail, the forget gate erases the unneeded information in the cell state. And the new input x_t of the current LSTM cell supplements information to generate a new cell state through the input gate. Then the new memory is used to produce the hidden states of the LSTM cell by the output gate.

4.6.2.2 Gated Recurrent Units (GRU)

GRU is another popular RNN with a gated mechanism. The GRU cell includes a reset gate and an update gate [138] to process the current input x_t and previous GRU cell's information h_{t-1} . The reset gate decides how much to forget in h_{t-1} . This mechanism enables the model to determine the importance of each input to the current state, aiding in making predictions. The update gate decides how much of the past information in h_{t-1} should be passed into the current GRU cell, which is essential for capturing long-term dependencies. These two gates are

beneficial to solve the short-term memory problem and feature with fewer parameters than LSTM.

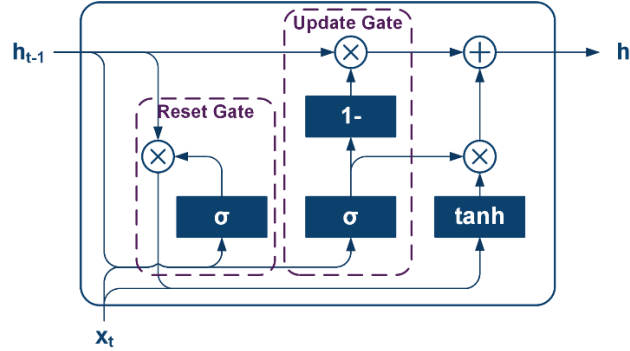


Figure 4-20 Schematic of GRU. x_t is the input at the current time frame. h_t is the hidden state. σ is the *sigmoid* activation function.

4.6.2.3 Attention Mechanism (AM)

The AM is a weighted sum based on the individual query, key and value vectors, converted from vectors of h_Q , h_K and h_V separately in the following equations [44]. In practice, the attention function is calculated on a set of query vectors simultaneously, which could be represented by:

$$Q_{attn} = H_Q W^Q = [h_Q^1, \dots, h_Q^{n-1}, h_Q^n] W^Q, \quad (4-42)$$

where n is the number of vectors, W^Q is the learnable parameter performing linear projection to get query vectors.

The key and value vectors are also packed together by:

$$K_{attn} = H_K W^K = [h_K^1, \dots, h_K^{m-1}, h_K^m] W^K, \quad (4-43)$$

$$V_{attn} = H_V W^V = [h_V^1, \dots, h_V^{m-1}, h_V^m] W^V, \quad (4-44)$$

where m is the number of vectors, W^K and W^V are learnable parameters performing linear projection to get key and value vectors.

The attention context is represented by:

$$H_{attn} = \text{Attention}(Q_{attn}, K_{attn}, V_{attn}) = \text{Softmax}(Q_{attn} K_{attn}^T) V_{attn}, \quad (4-45)$$

where H_{attn} is the resulting attention context.

4.6.3 Implementation Details

In this research, the prediction model is aimed at performing trajectory prediction up to 4s with an observation of past 2s. The training is deployed on the Nvidia GPU A100 at the High-Performance-Computing (HPC) platform.

In the model training, Adam is used as the optimizer, with a batch size of 128 and a learning rate of 0.001, respectively. The proposed method is trained for 60 epochs. The dropout rate is set to be 0.2.

The model is implemented using PyTorch. The root mean square error (RMSE) is used to calculate the regression loss during training:

$$RMSE = \sqrt{\frac{1}{p} \sum_{i=1}^p \left[(x_{t+i} - x_{t+i}^{gt})^2 + (y_{t+i} - y_{t+i}^{gt})^2 \right]}. \quad (4-46)$$

4.7 Evaluation Metrics

Two widely used measures of effectiveness are employed for performance evaluation, shown in (4-47) and (4-48), with lower values indicating better performance.

Final Displacement Error (FDE) [15]: This metric calculates the L2 distance between the predicted trajectory end point and the ground truth end point:

$$FDE = \frac{\sum_{i=1}^N \sqrt{(x_{t+p} - x_{t+p}^{gt})^2 + (y_{t+p} - y_{t+p}^{gt})^2}}{N}, \quad (4-47)$$

where x_{t+p} , y_{t+p} represent the predicted 2D position coordinates of a TA at the last time step ($t + p$) of the prediction horizon. The superscript gt represents that the data are from the ground truth, otherwise from the prediction. The final trajectory prediction point is related to the TA's driving intention, so it is important to evaluate its accuracy [15].

Average Displacement Error (ADE) [15]: It is used to measure the overall distance deviation of the predicted trajectory to the ground truth:

$$ADE = \frac{\sum_{i=1}^N \sum_{t=1}^p \sqrt{(x_{t+i} - x_{t+i}^{gt})^2 + (y_{t+i} - y_{t+i}^{gt})^2}}{N \cdot p}, \quad (4-48)$$

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

5 Parametrization of Road Scenes Based on Surveyed Boundaries

5.1 Abstract

Maps of road layouts play an essential role in autonomous driving, and it is often advantageous to represent them in a compact form, using a sparse set of surveyed points of the lane boundaries. While lane centrelines are valuable references in the prediction and planning of trajectories, most centreline extraction methods only achieve satisfactory accuracy with high computational cost, and limited performance in sparsely described scenarios. This chapter explores the problem of centreline extraction based on a sparse set of border points, evaluating the performance of different approaches on a public dataset, and proposing a novel method to extract the lane centreline by searching and linking the internal maximal circles along the lane. Compared with other centreline extraction methods producing similar numbers of centre points, the proposed approach is significantly more accurate: in our experiments, based on a public dataset of road layouts, it achieves a max deviation below 0.2m and an overall RMSE less than 0.01m, against the respective values of 0.4m and 0.1m for a popular approach based on Voronoi tessellation, and 2.6m and 1.2m for an alternative approach based on distance transform.

5.2 Introduction

Road-Layout (RL) maps are widely used in the field of autonomous driving and Advanced Driving Assistance Systems, as they provide accurate a priori knowledge of road geometry and lane markings, compensating for sensor inadequacies and uncertainties in the perception of the environment [139]. Because road features tend to follow regular geometric profiles – often straight lines – it is possible to represent these features with a sparse set of points with negligible loss of accuracy. Sparse representations of RL maps have many advantages in real-world applications: a lower volume of data is needed, and it is therefore more efficient to store, label and update a spare representation. This is particularly important, since the labelling and updating processes cannot be

reliably automated and often require a few manual steps [140] [27]. Also – and crucially – a smaller amount of data allows for faster access and processing of the maps by path planners, visualisation tools, and other relevant software components [27] [141].

In many applications of RL maps, it is useful to calculate the centrelines of individual lanes, because they provide reference trajectories and road-geometry information that are essential for the efficient implementation of trajectory planners. The centrelines are so important that they are often used in the definition of RL maps [142] even though they do not usually correspond to observable features on the road. Some RL map formats (like lanelet2) represent road boundaries, and to convert these to formats represented by road centrelines (like OpenDrive) it is necessary to extract centrelines from road boundary data [24].

5.2.1 Related Work

So far, a variety of methods have been proposed for road centreline extraction: some are based on the processing of aerial photography, some are based on the direct processing of road images, often taken from an aerial perspective, some are based on the skeletonisation of partitioned road polygons, and some are based on the skeletonisation of binary road maps.

When aerial images are available, road centreline extraction can be carried out using computer vision techniques, either via a two-stage process or an integrated procedure. In the two-stage methods, road regions are first detected through segmentation [143] [144] [145], then road centrelines are extracted; the techniques for the latter stage include morphological thinning [146] [147] and learning-based algorithms [148] [149]. With two-stage approaches, the accuracy and completeness of the extracted road centrelines are highly dependent on the accuracy of the road segmentation in the first step [149]. To mitigate this, integrated end-to-end methods using neural networks have been developed [149] [150]. However, these methods are typically designed to extract high-level geometric information of road networks, and they do not provide the level of detail and accuracy required to plan trajectories useful for automated driving [151].

Compared to aerial photography, panoramic-view images captured by vehicle-mounted cameras can better describe local road scenes. Deep learning-based (DL-based) methods, like transformers, can be used to estimate centrelines directly from the camera images [152].

However, these image-based centreline estimations can suffer from the limitation of camera performance, especially in bad weather or road surface conditions [153]. Further, extra work is needed to capture the global connectivity of lane paths and merge local maps into a global one [154], to compensate for the limited perception range of the cameras.

Though DL-based methods have gained much popularity, the training of models relies on the ground truth of centrelines, which cannot be produced as accurately as boundary polylines by DL-based technologies [155] [156] and relies heavily on manual annotation. Though some open-source datasets, like Argoverse [8] and nuScenes [10], provide centreline information, no information is provided about the methodology and the accuracy of centrelines.

Skeletonization techniques have greater potential to produce accurate centrelines for both automated driving and centreline ground truth. These approaches can benefit from the road boundary information, which is easy to obtain because boundaries are represented explicitly by visual and physical dividers, like lane markings and curbs. Prior work in this area can be categorized as contour-based and binary-map-based algorithms [157].

Contour-based algorithms rely on contour partition: the description of the target object is converted into a closed polygon composed of separated segments and scattered contour points. Several methods can then be used to extract the centreline from the resulting polygon, but the most popular solutions rely on Voronoi Tessellation (VT), i.e. the construction of a Voronoi Diagram. This technique decomposes the polygon into connected Voronoi cells, the centres of which are separate contour points. Each edge of a resulting Voronoi cell is a fragment of the perpendicular bisector of two contour points and, if the two contour points are opposite to each other, i.e. not adjacent, this fragment intersects the skeleton of the polygon [158]. Taking advantage of this property,

the Voronoi edges have been successfully used as an estimate of the centreline of rivers and roads [159] [160] [161]. Delaunay Triangulation, which is considered the mathematical dual of the Voronoi Diagram [158], can also be deployed in centreline estimation. The triangulation process decomposes the polygon domain into triangles, satisfying the criterion that no other contour points are inside the circumcircle of the newly built triangle. Then, the centrelines can be approximated by either linking the centroids of Delaunay triangles or midpoints of the edge shared by two adjacent Delaunay triangles [158]. The quality of the estimated centrelines through these approximations is highly dependent on the distribution of contour points. If the contour points are scattered in an uneven dispersion or a low density, the resulting centrelines tend to follow a zigzag trajectory around the real centrelines. Point intensification could be applied to the contour to increase their spatial resolution and thus produce a better approximation of the centreline, at the expense of increasing the computational cost. Furthermore, a smoothing process is usually needed to improve the smoothness of the estimated centrelines, which may cause some displacement from the real centre points [160].

Among binary-map-based algorithms, thinning methods are popular. Usually, these methods iteratively remove contour pixels according to pre-defined criteria of pixel deletion, which examine the relationship between the scoped pixel and its neighbourhood, until a skeleton of one-pixel width is obtained [162]. These thinning methods could be further classified into sequential and parallel algorithms, according to whether individual pixels can be processed dependently within the same iteration [163]. To improve the skeleton connectivity and erosion accuracy of thinning, hybrid thinning methods, combining different subclass approaches, are proposed [164]. At present, thinning methods are widely applied in pre-processing of pattern recognition and centreline extraction of biomedical images [165]. However, these methods usually suffer from excessive erosion of slanting patterns. And a trade-off should be made between the preservation of objective shapes and the production of one-pixel-width skeletons [166].

Instead of iteratively removing contour pixels to achieve the skeleton, distance transforms (DT) could be applied to the binary maps as a pre-processing procedure to obtain a coarse skeleton directly. DT calculates the distance from all background pixels to their closest object contour pixels based on the given binary image. The most common metrics used in distance calculation include Euclidean, Manhattan and chessboard distance, among which Euclidean distance is invariant to rotation and thus fits most applications [167] [168].

From the distance map produced through DT, a coarse representation of the object skeleton can be created by assembling local maxima. However, the resulting set of cells is not necessarily thin enough or well-connected to represent an accurate skeleton. Skeleton patterns can be sharpened by steepest ascent methods (SAMs), as demonstrated in several solutions for ridge detection [169]. Based on this idea, [170] identified so-called saddle points among local maxima in the distance map and connected them by searching non-local maxima in-between along the steepest uphill path, which produces connected and thin skeletons.

Another branch of binary-map-based methods that has received limited attention so far, relies on the identification of the centres of the maximal inscribed circles of a polygon, defined as Maximal Disks (MDs) [171]. [172] calculated a signed sequential Euclidean distance map based on the outline of the skeletonization target and then ran through all pixels within the outline to identify MD centres. According to the MD principle, if a selected pixel is the MD centre, the distances from this pixel to its two nearest contour pixels should be equal. However, due to discretization, such pixel could not be always found. So, two distances are calculated instead: the distance from the selected pixel to its nearest contour point, and the distance from one pixel of its eight neighbours to its nearest contour point. Then a connectivity criterion based on these two distances is proposed to determine the true MD centres. Unlike the thinning method, no iteration process is needed so MD is more efficient. To avoid generating redundant branches from the skeleton, [173] introduced a contour partition process, based on the discrete curve evolution, before exploring MDs. Through partition, the integrated contour

is divided into several individual segments according to the pattern topology. Then MDs are generated in the way that their touching points with the contour locate in different segments, whose loci are the main skeleton of the pattern with no spurs.

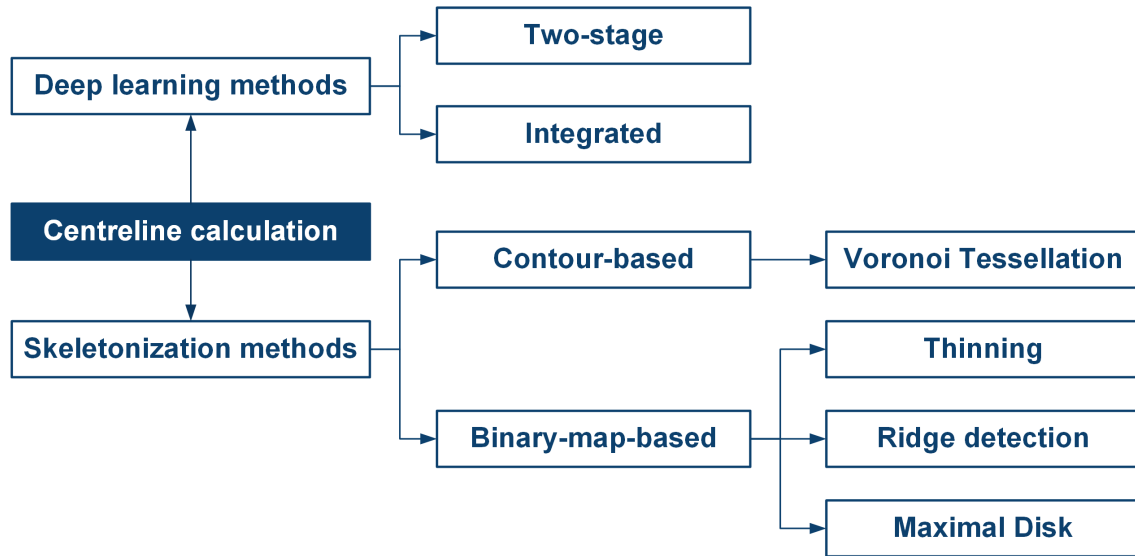


Figure 5-1 Classification of road centreline calculation methods

An overview of the classification of road centreline calculation is shown in Figure 5-1. In general, a more accurate skeleton can be obtained as the resolution increases in the distance map; however, the implementation is usually computationally prohibitive, as confirmed by the results presented in Section 5.4

5.2.2 Motivations and Contributions

The analysis of prior work shows that the methods for the extraction of high-accuracy centrelines from a sparse representation of the road boundaries tend to rely on spatial resolution enhancements, which further increase the computation and storage burden. The aim of this chapter is to propose a solution to calculate the minimal set of points that accurately describes the centreline of road lanes in the sparsely defined scenarios, without changing the representation through re-sampling, interpolation, discretisation or filtering. This approach enables a compact description of RL maps, efficient usage of computation and memory resources, and fast labelling of aerial views.

The novel contributions of this chapter are: constraint componets

1. A novel approach for the extraction of the road centreline is proposed, based on MD. Instead of re-sampling the space, MDs are formed directly based on the constraining elements, either segments or their extreme points taken from the boundaries.
2. In addition to the identification of the relevant MDs, a criterion is proposed to link the centres of the MDs into a connected centreline.
3. To relieve computation burden, a segment pairing method to assemble suitable constraining elements is also presented.
4. Performances of the proposed approach and two popular skeletonization methods, based on VT and DT, are compared in sparse and dense scenarios generated from a public dataset of road layouts.

This work is significant to the robotics, mapping, and transportation systems communities because:

1. By producing a minimal set of centreline points, the proposed method reduces the size of RL maps, is memory-efficient, and, while its computational cost is comparable to other skeletonization methods, the reduced amount of data can save computational resources down the processing pipeline, typically in trajectory planning applications.
2. The proposed method can be used to convert RL maps from border-based to centreline-based formats, retaining only the significant data points. This simplifies the manual maintenance of the produced RL maps.
3. This chapter provides valuable data on the accuracy, the computational cost, and memory usage of the three methods mentioned above (MD, VT, and DT), allowing researchers to select the best approach when performing centreline extraction in dense and sparse scenarios.

In the rest of this chapter, Section 5.3 presents the centreline extraction method based on MDs, as well as the proposed implementation of the solutions based on VT and DT. Experiments and results are reported and discussed in Section 5.4, followed by the conclusions drawn in Section 5.5.

5.3 Methodology

A centreline extraction method based on the MD principle is proposed in this section. The process consists of two steps: calculating centre points based on MD and then linking centre points to form a centreline. The details of the method are elaborated on in the following paragraph.

5.3.1 The Extraction of Lane Centre Points Based on MD

In a road map defined by sparse border points, a road lane can be partitioned into a set of connected lane pieces, each of which consists of 2 opposite line segments and their extreme points. A lane example is illustrated in Figure 5-2.

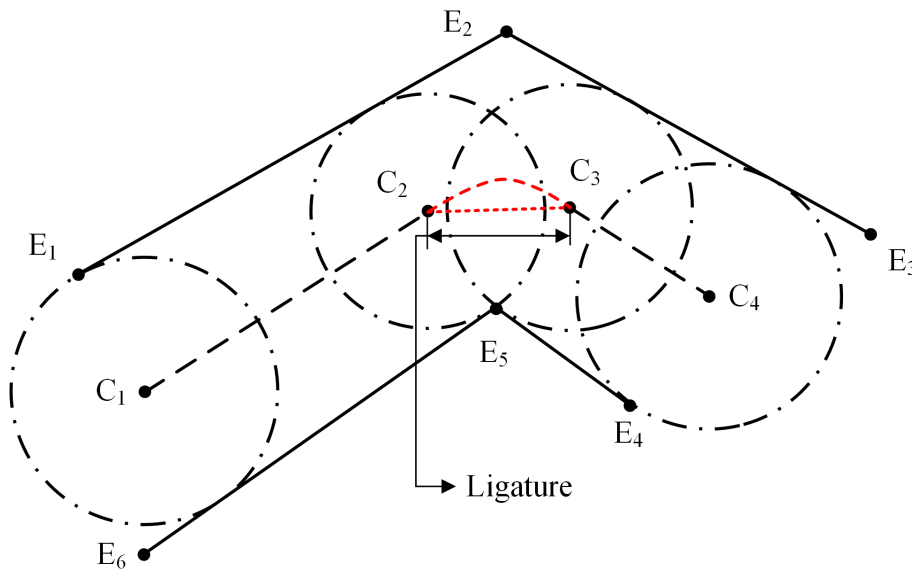


Figure 5-2 Illustration of a combination of two-lane pieces. E_i are extreme points of the line segments. C_i are circle centres of the maximum circles. The red straight dotted line denotes the ligature. And the curved dashed red lines are estimated centerlines in the ligature.

According to the MD principle [171], the lane centreline extraction is converted into finding and linking the centres of the maximal inscribed circles. A sketch of a maximal circle is the circle centred at C_1 , which passes through one extreme point and is tangent to one of the opposite line segments. C_1 is equidistant to both of the opposite lane border segments that it touches, and thus is on the bisector of the segment pair. So is C_2 , and all centres of the maximal circles touching these

two segments lie on the same bisector segment defined by C_1 and C , which is also the centreline. Taking advantage of this property, it is possible to reduce the amount of data used to represent the centrelines of a simple lane to only two points, without loss of accuracy. Unfortunately, this approach cannot be applied directly in the presence of sharp bends and complex lane structures, so the aim of this chapter is to identify the minimum number of MDs that can be used to represent the lane centrelines in any sparsely described scenario.

When considering sharp bends that are sparsely described, as shown in Figure 5-2, it is important to notice that the centreline might rotate around one of the extreme points, so an infinite number of MDs is needed to accurately represent this line. This complex curve cannot be described by a straight segment, leading to some loss of accuracy in any centrelines represented by a finite number of MDs. This approximation is noted as ligature [171], and it is shown as a red dotted line in Figure 5-2.

To mitigate this problem and improve the accuracy of the centreline, more cases of inscribed circles are explored. An illustration of how inscribed circles can be related to surrounding lane segments is given in Figure 5-3.

According to the number of segments and points constraining the MD, four basic cases can be identified, shown as (a)-(d). In these basic cases, the segments and the points constraining the MD are not related, but there are also two special cases, shown as (e) and (f), where one of the constraining points lies on one of the segments.

Of all the six cases, (f) is the most common one in road scenarios, as it is characteristic of the beginning of a new lane piece with a slight bend. All other cases contribute to the alleviation of the direction inconsistency at sharp bends.

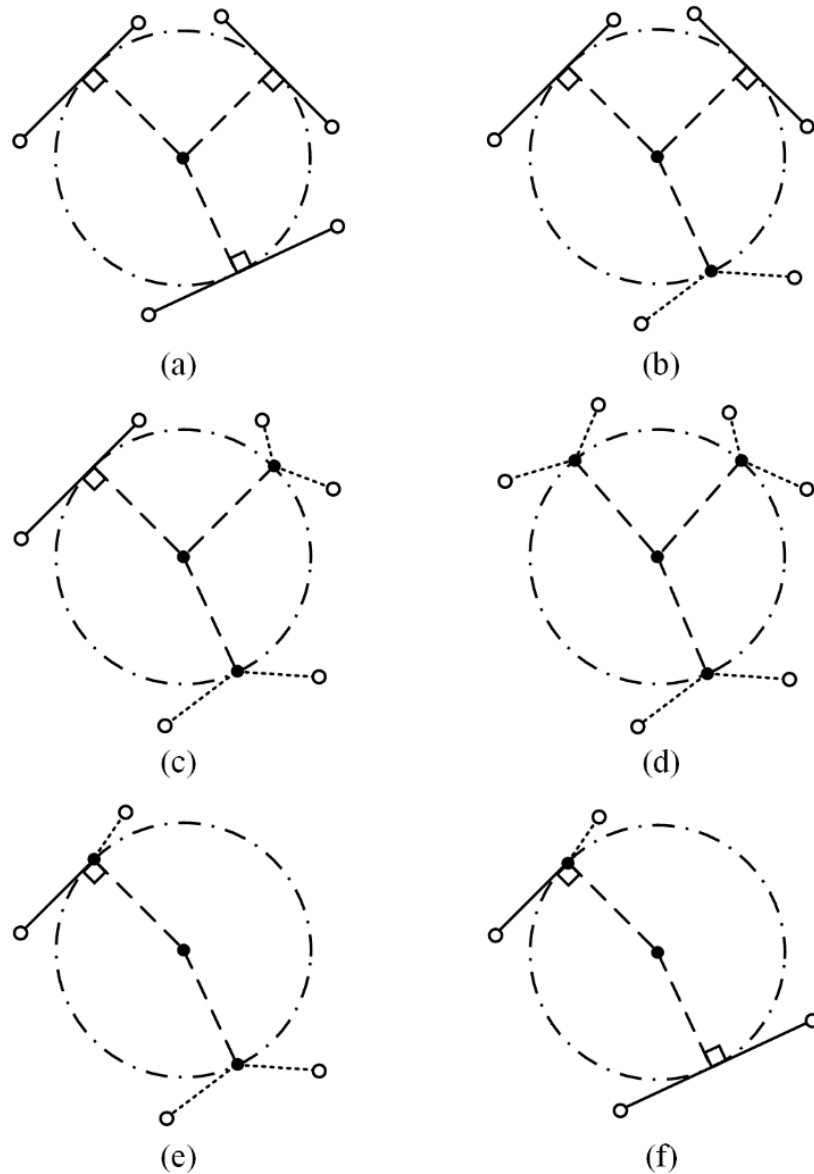


Figure 5-3 Illustration of maximal inscribed circle types. Two types of constraints: the circle being tangent to a segment or passing by a point. In (a), the circle is constrained by 3 segments. In (b), the circle is constrained by 2 segments and 1 extreme point. In (c), the circle is constrained by 1 segment and 2 extreme points. In (d), the circle is constrained by 3 extreme points. In (e) the circle is constrained by 1 segment and 2 extreme points, with one of the two points lying on the segment. In (f), the circle is constrained by 2 segments and 1 extreme point, which lies on one of the two segments.

5.3.2 Segment Pairing

The previous section explains that at least two separate border segments or some of their extreme points are required to constrain an MD, so it is possible to identify the MDs by searching for suitable sets of segment pairs. It is important to notice that at least two of the three border constraints (either segments or extreme points) have to be associated with separate boundaries, i.e. either the right or the left border, otherwise they are constraining a cove, which is irrelevant for centreline calculations. It would seem reasonable to assume that segment pairs are somehow “overlapping” with each other, meaning that they are vaguely in front of each other. However, as Figure 5-3 shows, this relationship is not always straightforward. A thorough segment pairing can be achieved, for example, by selecting one segment from one border and exploring all combinations by matching it with two distinct segments in the counterpart. Though this pairing is easy to realise, time consumption will surge in the following steps for calculating MDs because most of the segment pairs will produce MDs interfering with lane boundaries, some of these cases being listed in section 5.3.3. To overcome this problem and avoid a computationally expensive search across the entire map, this chapter proposes to search for neighbouring segments.

The search starts with the selection of an extreme point of a segment. A circle centred at this point is formed with a fixed customized radius and all points within the circle, on the opposite border, will be considered as potential constraining points for an MD, or extreme points of a constraining segment. This can be seen as a range search procedure. The result of a range search is shown as all points inside the circle centred at R_3 in Figure 5-4.

Some of these identified points are unsuitable for matching, because belonging to a section of the border that is unreachable by an MD. This situation is common in a U-type lane, as shown in Figure 5-4. An in-lane MD could not be found under the constraints of R_3 and L_8 or their relevant line segments. So L_8 should not be matched with the selected extreme point R_3 . To remove these unwanted points like L_8 , accumulated distances along the border are considered as the point filtering criteria. The circle points inside the search range are first grouped by their

indexes, in which the point indexes are numerically continuous and increase from the lane start to the end in sequence. For example, R_3 -relevant points will be divided into two groups, one containing L_3 , L_4 and L_5 and the other containing L_8 only. Then, the group of points with smaller indexes will be chosen if the accumulated distance of the selected extreme point to the start is smaller than its distance to the end; otherwise, the group of points with bigger indexes will retain. As Figure 5-4 shows, the extreme point R_3 is closer to the start point R_1 and the group containing L_8 , having a larger index, is discarded.

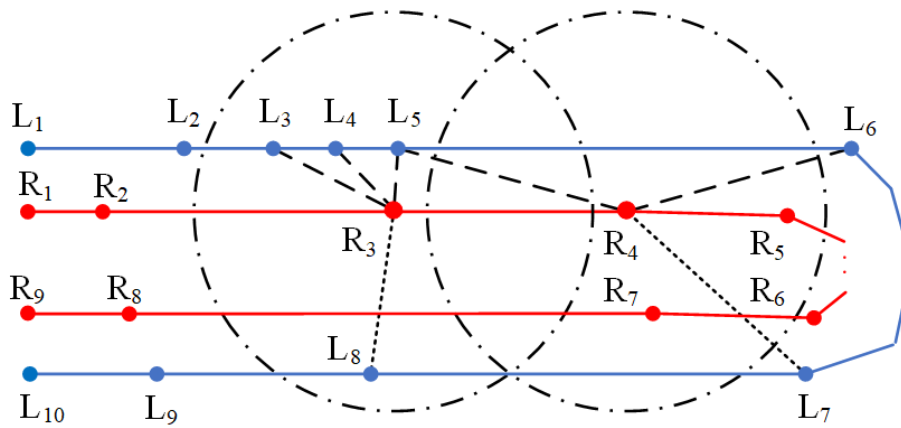


Figure 5-4 An illustration of a range search and a nearest-neighbor search, based on point R_3 and point R_4 separately. Blue line segments belong to the left lane border while red ones belong to the right lane border. In the range search, R_3 should be matched with L_3 , L_4 and L_5 , but not L_8 . In the nearest neighbor search, R_4 should be matched with L_5 , and L_6 , but not L_7 .

With the matched points, segment pairs could be established by constructing segments according to the point indexes.

Due to the sparse distribution of the points, no opposite point may be found within the given circle in the range search. But a point pair does exist. An example could be seen in Figure 5-4 that no points are found in the circle centred at R_4 but an MD might exist with constraints of points R_4 and L_6 and their relevant segments. To avoid this omission, a nearest-neighbour search is performed after the range search to supplement its limitation on the search scope. All points on both borders are stored in the structure of a 2-dimensional tree for easy and quick point query.

The number of nearest neighbours is set to 3 to make sure the correct and efficient point pairing. And the same point filtering method as in range search is used to remove points failing to constrain an in-lane MD.

5.3.3 Circle Centres Filtering

Through segment matching and circle centre extraction, all possible circles are found. But some circles interfere with the lane borders (like C_1 in Figure 5-5.), or the contact points are not on the border segments but their extension (like C_2 in Figure 5-5.). These circles have either exceeded the domain of the lane or overstepped the constraint of their related segment pairs. Thus, those circles which are not completely inside the lane polygon, or are weakly constrained by their related segment pairs, are deleted.

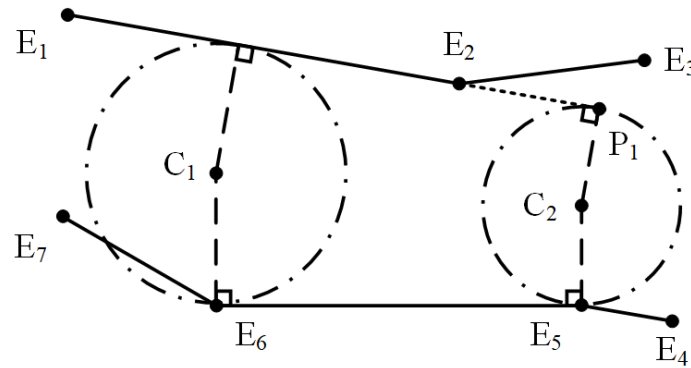


Figure 5-5 Example of MDs that need to be filtered out. The circle C_1 is crossed by border segment E_6E_7 , thus is beyond the lane region. The contact point of the circle C_1 with segment E_1E_2 is on the segment extension, which has produced a circle beyond the scope of segment constraint.

5.3.4 Connectivity of Circle Centres

The previous section describes how to select a minimal set of MDs, whose centres are lying on the centreline. The centreline, however, is defined as an ordered sequence of points, so it is necessary to organize and connect the obtained circle centres.

As illustrated in Figure 5-3, the MDs touch a segment through contact points, which are either tangent points on the segment or its extreme points. The position of these contact points can be used to sort the MDs to form a centreline.

At the beginning of the connecting process, the MDs are sorted according to the contact points on a chosen border, based on the border-along distance between the contact point and the start point of the border. For example, in Figure 5-6, C_3 comes earlier than C_4 because its contact P_1 point is closer E_1 , the start point of the segment, than P_2 .

In case of a tie (typically, two MDs passing by the same extreme point), the MDs are sorted according to the contact points on the opposite border, based on the border-along distances to the start point of the new border.

It is not unusual for some MDs to touch one of the borders with two projection points, like C_1 , C_5 and C_6 in Figure 5-6. In these cases, the projection point that is closer to the border start point participates in the sorting. For example, in Figure 5-6, C_1 comes earlier than C_2 because the contact point T_{11} has a shorter border-along distance than T_2 .

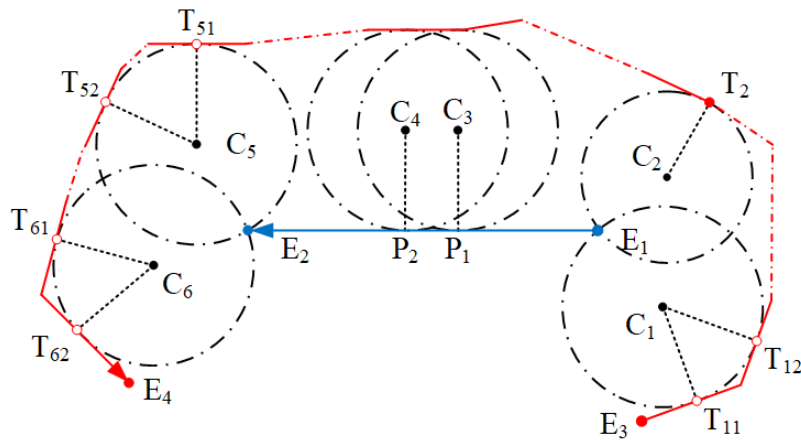


Figure 5-6 The left lane border, in blue, and the right lane border, in red, are oriented according to the direction of the arrows. The associated MDs are shown together with their contact points.

5.3.5 The Extraction of Lane Centrelines Based on VT

VT is one of the most established solutions in centreline calculation and is chosen by most papers. Better performance is achieved with modifications, but they are only relevant to specific scenarios. Because of this, this section only considers the original approach of VT. The procedure for centreline estimation based on VT is inspired by Golly et al. [160], but with some changes: the dead-end branches are filtered by the shortest path search between the start and end nodes instead of by counting Voronoi edge connections, and no further smoothing is performed on the Voronoi edges. Specifically, the steps below are proposed:

1. Define a closed lane polygon by linking scattered lane border points sequentially.
2. Perform Delaunay Triangulation inside the lane polygon [174].
3. Operate VT [158].
4. Filter and prune the Voronoi edges.
5. Chain Voronoi edges into one consistent line.

Lane polygons are usually nonconvex and, as a result, Delaunay triangulation will be performed both inside the lane polygon and in its concavity, as shown by the dashed triangles in Figure 5-7. These triangles should be removed because they do not belong to the lane region. The result of this operation is shown in Figure 5-7 by the solid blue lines.

VT is performed after triangulation. The resulting Voronoi edges are represented by the grey lines and the red lines in Figure 5-7. The grey lines are edges perpendicular to border segments, the lane entrance, or the lane exit. And, because they do not contribute to the construction of the centreline, they need to be filtered out. After this filtering, there might still exist some spurs, like the dashed red line in Figure 5-7. To remove these spurs and achieve a well-connected centreline, the start and end nodes of the centreline are firstly identified by searching for the intersection points of the Voronoi edges and the predefined entrance and exit segments (like the nodes N_1 and N_3 in Figure 5-7). Then, a unidirectional path search is performed to find the path connecting the start and end nodes through the remaining Voronoi edges. In this way, edges

belonging to the centreline will be connected in sequence, and the spurs will be removed. The output centreline is shown as the red line segments in Figure 5-7.

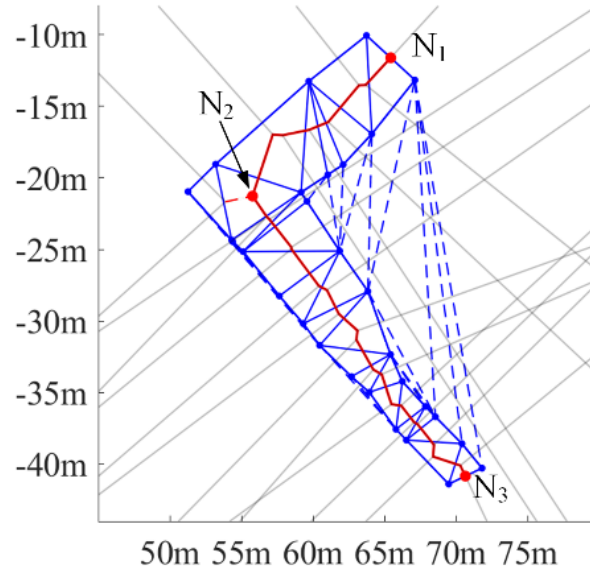


Figure 5-7 An illustration of the centreline (solid red segments) estimated by the VT. The grey lines are Voronoi edges perpendicular to border segments. The solid blue segments are the resulting edges from inside-lane Delaunay triangulation, while dashed blue segments are edges of Delaunay triangles outside the lane polygon. N_1 and N_3 are the start and end nodes of the centreline separately, and N_2 is a branch node. The branch node has three connected edges, one of which is a residual spur (the red dashed line).

5.3.6 The Extraction of Lane Centrelines Based on DT

The proposed centreline extraction method, based on DT, can be organized into four main steps:

1. Project lane boundaries on a binary grid of appropriate resolution.
2. Perform 8-neighbour Euclidean Distance Transform inside the lane polygon [167].
3. Operate ridge detection with 8-neighbour Discrete Laplace Transform [169].
4. Find and connect centreline points by Dijkstra's Searching [175].

In the first step, the lane boundaries are projected on a binary grid by assigning a value of 1 to the cells intersecting with them, and a value of 0 to the non-intersecting cells. The entrance and exit of the lane polygon are not projected to the binary grid to avoid meaningless values of distance that cause a bifurcation of the centreline, as shown in [161].

A distance map is then calculated from the binary map, by propagating the distance from the boundary to the neighbouring cells, across the whole map [167]. The metric of 8-neighbour Euclidean Distance is used to provide a smooth variation of the distance value. To reduce the computational load, the DT is restricted inside the lane polygon with a created in-lane discriminant matrix. Furthermore, the distance between two orthogonally adjacent cells is set to 1 instead of the resolution of the grid map. This means that the intensity of the ridge in the ridge detection will not change with the resolution. The resulting distance map is shown in Figure 5-8 (a). The coloured area is the area of interest, and the different colours represent the distance values derived from DT.

A process is then applied to sharpen the several-pixel-width central domain: the Discrete Laplace Transform (DLT) is used to obtain a ridge map by evaluating the second-order difference between the cell of interest and its neighbours in the distance map [169]. The 8-neighbourhood discrete Laplacian of a cell is used in this research.

Figure 5-8 (b) illustrates the results of the DLT. The central area has shrunk to a line, exhibiting a several-pixel width. At the same time, some secondary ridges have emerged in the transverse direction, due to the discrete nature of the computation.

To find the final one-pixel-width centreline points and to connect them in order, Dijkstra's search is deployed. Selecting and linking the centre points is done in conjunction with the Dijkstra's search, by tracking the local minima of the ridge map, corresponding to the location of the ridge. Since Dijkstra's search is typically designed to operate in the positive domain and the ridge map is mostly negative, a new map is calculated by adding the absolute value of the global minimum to

the value of each cell in the ridge map. The cost function is set to find the minimal accumulated values of the resulting map.

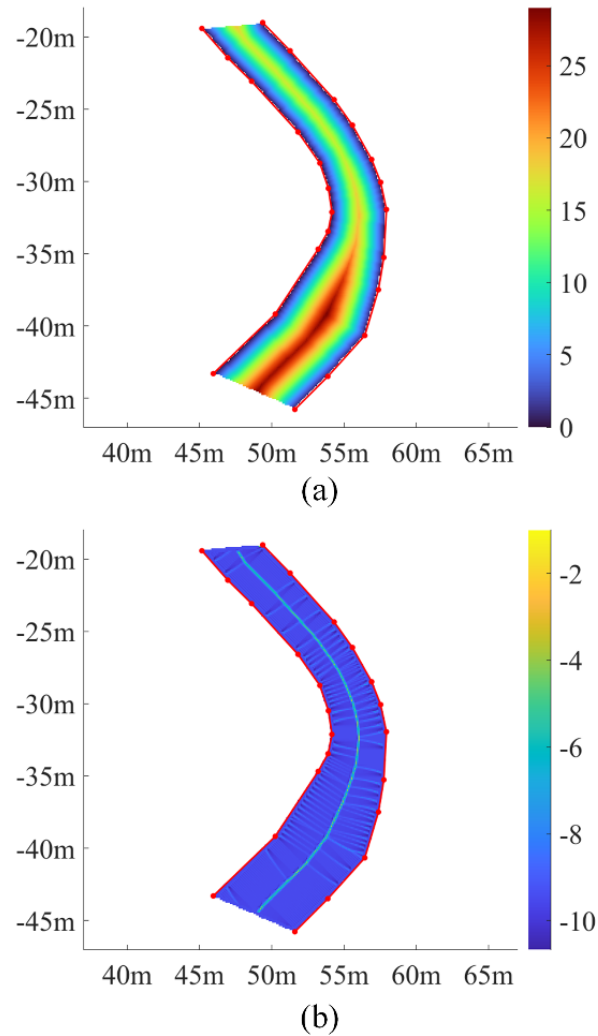


Figure 5-8 Visualization of the results of the intermediate steps of centreline extraction based on DT. The lane border segments are marked in red. (a) is the distance map and the colours represent distinct distance values from the pixel to its nearest border segments. (b) is the ridge map and colours represent the ridge intensity.

5.4 Experiments and Results

The centreline extraction results based on Maximal Disks (MD), Distance Transform (DT) and Voronoi Tessellation (VT), are compared in the following section in terms of computational efficiency and performance accuracy.

5.4.1 Data Description

To evaluate the proposed method for centreline calculation, the target road lanes are extracted from the maps included in the 'inD Dataset' [20], which are represented by sparse points. The map used for this evaluation, which is the first scenario in the dataset, is shown in Figure 5-9. It covers an urban intersection and includes both pedestrian paths and vehicle driving lanes.

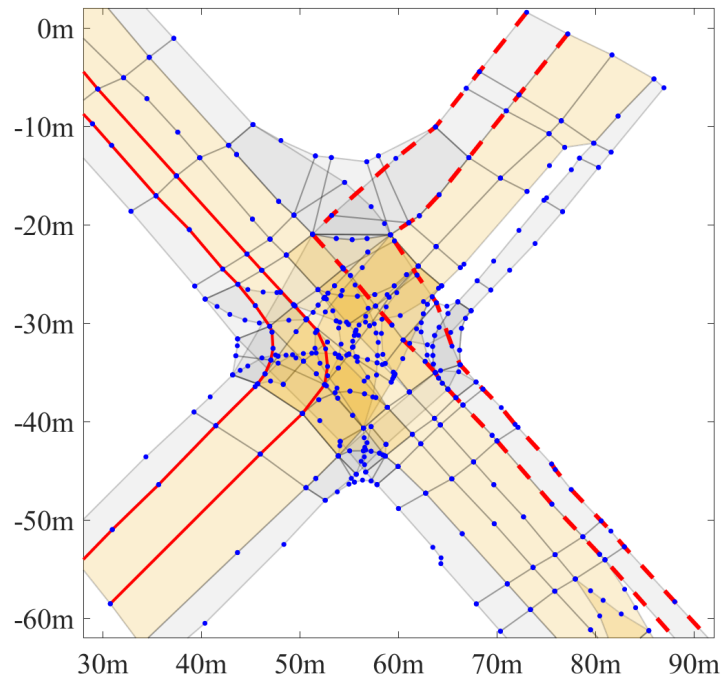


Figure 5-9 The first scenario in the inD. Sparse points are marked in blue. Lanelets are represented by polygons. Vehicle driving lanes and non-driving lanes (like pedestrian paths) are marked in light yellow and grey separately. Examples of a complete pedestrian path and a driving lane are bounded by solid and dashed red lines separately.

In the map, geometric information like point coordinates is presented in the Lanelet2 format [139] and a lane is defined as a series of connected lanelets, which are the basic elements in the map and are modelled by polylines. Generally, the point density is low on the straights and it is high on the corners. A total of 28 complete lanes are extracted from the map, covering around 2844 m.

5.4.2 Experimental Results and Comparisons

All methods are programmed and simulated using MATLAB software, on a computer with an AMD R7-4800H processor and 16GB RAM. It is important to note that the implementation of the MD approach is based entirely on custom code, which offers plenty of opportunities for optimization, while the VT and the DT take advantage of very mature libraries from Mathworks (delaunayTriangulation, voronoi, bwdist, conv2, shortestPath).

In terms of computational efficiency, the total execution times (ET) of centreline extraction of different methods are recorded. For a fair comparison of individual approaches, the time required by data loading and storing processes is ignored. Each script is run multiple times and the median of the ETs is selected. The resulting ETs are illustrated in Table 5-1.

In order to evaluate the performance accuracy of each method, the maximal deviation to the benchmark and the root mean square error (RMSE) of all 28 lanes have also been calculated and shown in Table 5-1. The benchmark centrelines are extracted by DT with a resolution of 0.01m. Considering that centrelines extracted from different methods have different resolutions, the accuracy is evaluated after sampling each centreline with a fixed step of 0.005 m.

Generally, MD is fast in execution and produces accurate centrelines, with a maximal deviation of 193.90 mm and an RMSE of 9.54 mm through all 28 lanes. Most importantly, it provides the most compact representation of centrelines with 2301 centre points, compared to methods based on DT and VT which achieve comparable performance in terms of maximal deviation and RMSE (see results

highlighted in violet and yellow, respectively, in Table 5-1). By considering methods that calculate a comparable number of centreline points (highlighted in green in Table 5-1), MD produces much more accurate results.

Table 5-1 The Comparison of Results of 3 Methods

Method	ET [s]	Point Number	Max Deviation of all lanes[mm]	RMSE of all lanes [mm]
MD	7.07	2301	193.90	9.54
DT (0.9m)	1.27	2343	2625.00	1244.96
DT (0.4m)	3.26	5379	480.84	101.41
DT (0.2m)	11.71	10770	155.58	47.94
DT (0.1m)	43.93	21575	85.00	22.74
VT (Sparse)	0.33	1355	5573.86	854.46
VT (3m)	0.45	2565	440.74	122.58
VT (1.5m)	0.69	4422	175.05	36.61
VT (1m)	0.91	6273	74.02	16.96
VT (0.2m)	3.72	28563	11.22	2.85
VT (0.1m)	7.12	56374	9.71	2.77

VT has the best performance in terms of time efficiency, even with a high resolution of 0.1 m. Starting from the original sparse border points, VT produces a poor centreline extraction with a huge maximal deviation from the benchmark centreline. Its accuracy improves with higher spatial densities. However, the number of centre points grows with it. With the point resolution of 1 m, the number of centre points is almost three times bigger than the output of MD, and it increases linearly with the resolution.

DT operates the slowest, and its execution time surges with an increased resolution. By doubling the resolution from 0.2m to 0.1m, the execution time increases by almost four times. Although the performance of DT improves with a higher resolution, it is computationally prohibitive.

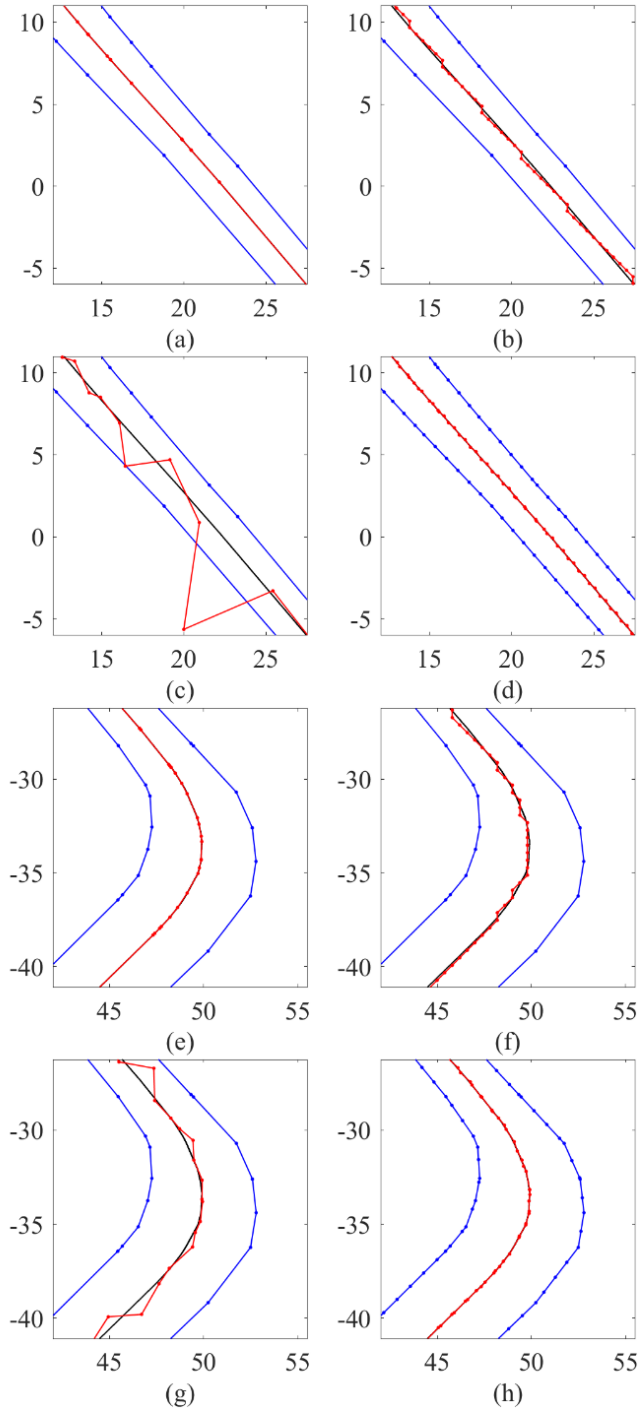


Figure 5-10 Illustration of centreline extraction results of one lane. (a)-(d) are results in the straight part and are obtained by MD, DT (0.4m), VT (original points) and VT (1m) separately. (e)-(h) results in the normal bending part and are obtained by MD, DT (0.4m), VT (original points) and VT (1m) separately.

To illustrate the performance of the 3 methods along the lane, the extracted centrelines are presented against their benchmark in Figure 5-10 and Figure 5-11.

MD and VT are evaluated on the original border points, to compare the performance of these two methods in sparse-point scenarios. Considering that DT only operates on a fixed resolution of the map, a grid of 0.4 m is selected to assess the performance of DT in low-resolution scenarios. Finally, the results of VT operating at a resolution of 1m are also shown, to examine the performance of methods with similar accuracy to MD. The results comparison is shown in Figure 5-10.

In both straight and bending parts, MD shows a tidy presentation of the lane centreline. The accuracy of centreline extraction is high as the extracted centrelines are well matched with the benchmark centrelines. DT provides a stair-like centreline in the straight lane and produces jags in the bending, which degrades the smoothness of the centreline. This is caused by searching in the discretized grip map.

The results from VT starting from the original sparse points are not smooth. It could be seen that where lane border points are not aligned well transversely, a protrusion occurs, which may even cut cross the lane polygon. In any case, there are visible deviations in the centreline, which are caused by the uneven distribution of lane border points. This problem can be mitigated by re-sampling the boundaries with a finer resolution, as shown by the performance of VT with borders sampled every 1m.

A closer look at a sharp corner is shown in Figure 5-11. This is the 19th lane in scenario 1 of the inD Dataset and has a near-90-degree corner. The max error of MD is always met in sharp corners like this one. From the picture, it is possible to see that the centreline extracted by MD is flat compared to the benchmark. This is the ligature mentioned in Section 5.3.1. MD performs an undercut in the sharp bending, underestimating the real centreline, because the method is not designed to track curved centreline. However, the start and end points of the ligature have been accurately calculated by MD, minimizing the deviation from the benchmark.

This deviation could be reduced by adding intermediate MDs in the ligature, but this solution will increase the execution time and the number of centreline points.

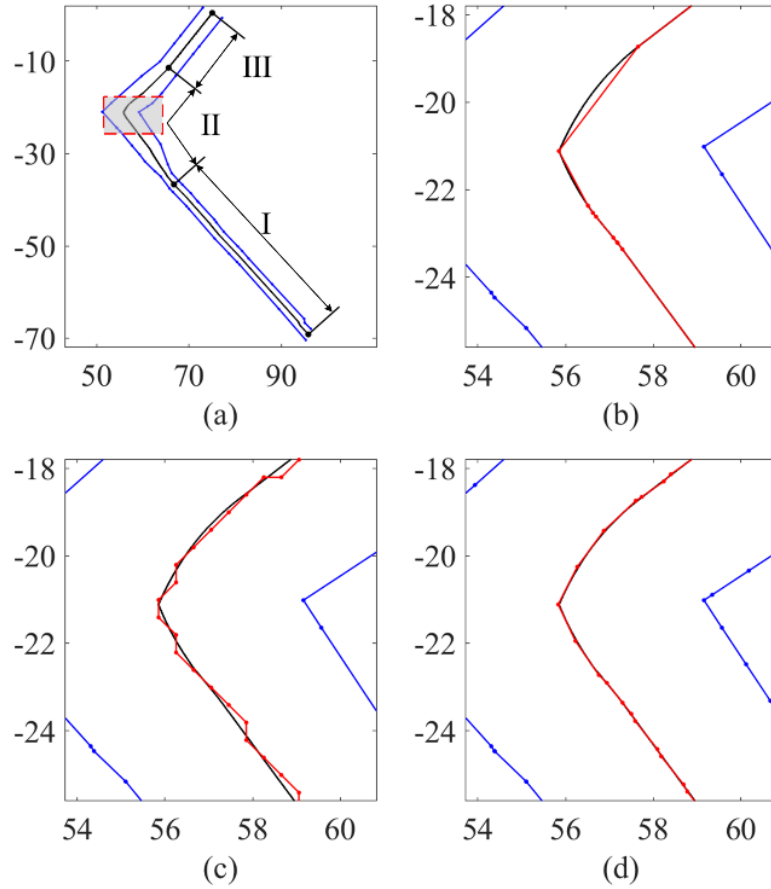


Figure 5-11 Illustration of centreline extraction performance in the sharp corner. (a) is the target lane and its centreline benchmark. (b)-(d) are results from MD, DT (0.4m) and VT (1m) separately.

To better illustrate the effect of the ligature, the selected lane is partitioned into 3 sections, as Figure 5-11 (a) shows, and the deviation along the benchmark centreline in these 3 sections is shown in Figure 5-12. The deviation of MD in the sharp corner is obvious in section II, reaching its maximum of 0.1939 m. It could be also seen that in the straight parts of the lane, MD produces a centreline very close to the benchmark while VT and DT produce centrelines fluctuating around the benchmark.

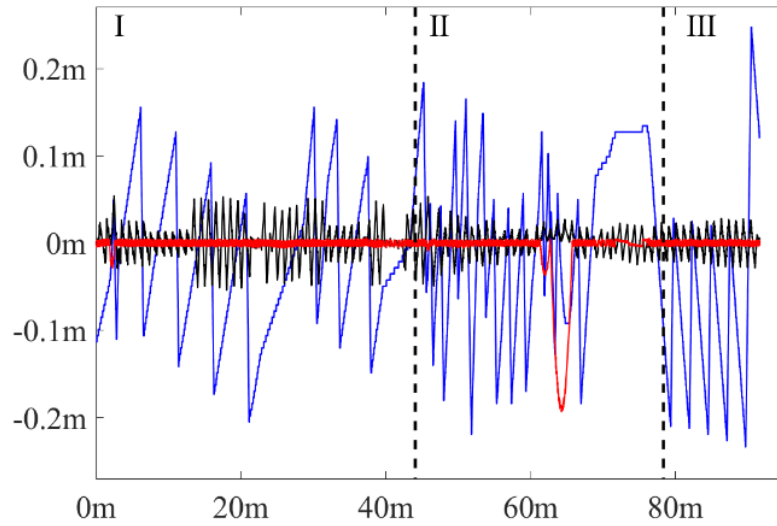


Figure 5-12 The deviation from the calculated centreline to the benchmark centreline. The red, black and blue lines are from MD, VT and DT separately.

5.5 Conclusions

This chapter presents a lane centreline extraction method based on maximal disks (MDs), which is designed for optimal performance in sparsely defined driving scenarios.

The proposed method features light data storage and high execution efficiency. Results have shown that:

1. A smooth and accurate centreline could be obtained using this approach.
2. The proposed approach achieves higher accuracy compared to other methods like DT and VT in sparse point scenarios or with small numbers of data points.
3. For the same target accuracy, the proposed method is significantly faster than the DT and slightly slower than VT. While the other two methods have been implemented with mature and efficient libraries, there is plenty of room for performance improvements in the implementation of the proposed approach.

Because of the minimal number of extracted centreline points, the proposed method can support efficient implementations of path planning algorithms. It can

also be used to convert RL maps between different centreline-based or border-based formats, without increasing the number of data points.

It is important to notice that centreline ligatures, which usually happen in sharp bends, greatly penalize the performance of the MD method. Future work will be focused on improving centreline extraction in these scenarios.

6 Trajectory Prediction Model based on LSTM and Agent Historical Status

6.1 Introduction

In recent years, the field of trajectory prediction has increasingly attracted the attention of researchers, primarily due to its critical role in the development of advanced autonomous driving systems.

This surging interest stems from the necessity to accurately foresee and understand the dynamic evolution of traffic scenarios in autonomous driving. Trajectory prediction is beneficial in enabling these systems to anticipate future positions and movements, thereby enhancing their decision-making capabilities and operational safety.

A significant aspect of this field is the reliance on data-driven methodologies, which have proven to be highly effective. These methods leverage extensive naturalistic driving datasets, learning from real-world scenarios to create models that can predict trajectories with higher accuracy and reliability. The advantages of such an approach are manifold. By directly analysing and extracting patterns from actual driving data, these models can develop a more subtle and comprehensive understanding of movement and behaviour of traffic participants. This in turn contributes to more reliable and efficient planning of trajectories, which is crucial in rapidly changing and unpredictable environments.

6.1.1 Related Research

Trajectory prediction models try to infer the future states of the target traffic participant by deducing its historical status. Several neural networks (NNs) could be used to infer the inter-relationships among data of consecutive time steps. One example is Convolutional Neural Networks (CNNs), which are predominantly employed in the early stage. This choice was largely influenced by their compatibility with the output of perception sensors.

In autonomous driving, the detection sensor data at each time frame are usually in the form of point clouds. CNNs are usually used to explore the spatial

relationships among these sparse sample points to identify separate entities. The details of trajectory prediction backbones based on CNN can be found in section 3.5.1.2. Considering CNNs' limitations, Recurrent Neural Networks (RNNs) have been identified as a more suitable alternative, particularly due to their inherent design which is better adapted to handle time series data. RNNs utilize memory mechanisms that are specifically tailored to maintain and transmit information through time, making them more adept at capturing temporal dependencies in data. Among the various implementations of RNNs, Gated Recurrent Units (GRU) [176] and Long Short-Term Memory (LSTM) [177] networks have gained prominence in trajectory prediction tasks. These models are particularly valued for their ability to handle the long historical status of target agents (TA) without vanishing gradients. More details of trajectory prediction backbone methods based on these RNNs can be found in section 3.5.1.3.

In general, RNNs have replaced CNNs in the trajectory prediction work. Both LSTMs and GRUs are deployed. And, the encoder-decoder structure can further improve the capabilities to do a summary of the historical status and infer the future trajectories of the target agents.

6.1.2 Motivations and Contributions

This work applies the encoder-decoder architecture and fills it with different RNN units, GRU and LSTM, to compare their performance. The RNN unit with better performance will be selected in the following research.

To determine the appropriate hyper-parameters of the RNN units, experiments are performed to study the influence of the hidden state sizes on the prediction performance. Also, the types of historical observations are also explored to study which information contributes more to the performance of the prediction model.

This work is significant to the trajectory prediction communities because it provides valuable data on the performance of the encoder-decoder prediction structures based on different input data and model settings, allowing researchers to make a well-informed decision while designing their prediction models.

In the rest of this chapter, Section 6.2 presents the trajectory prediction model. Experimental evaluation is performed in Section 6.3. Finally, the conclusions and discussions are in Section 6.4.

6.2 Prediction Model

The aim of this research is to predict future positions for TAs, which intrinsically is a regression problem. This is distinguished from classification tasks such as intention or behaviour prediction.

According to the findings in Chapter 3, TF achieves state-of-the-art performance as the prediction backbone. However, it faces over-fitting issues in our experiments. Given that LSTM is also competitive in trajectory prediction, it is chosen as the backbone. Instead of a single LSTM, an encoder-decoder framework is used. This framework is a common approach and is modified from the [176]. The model is shown in Figure 6-1.

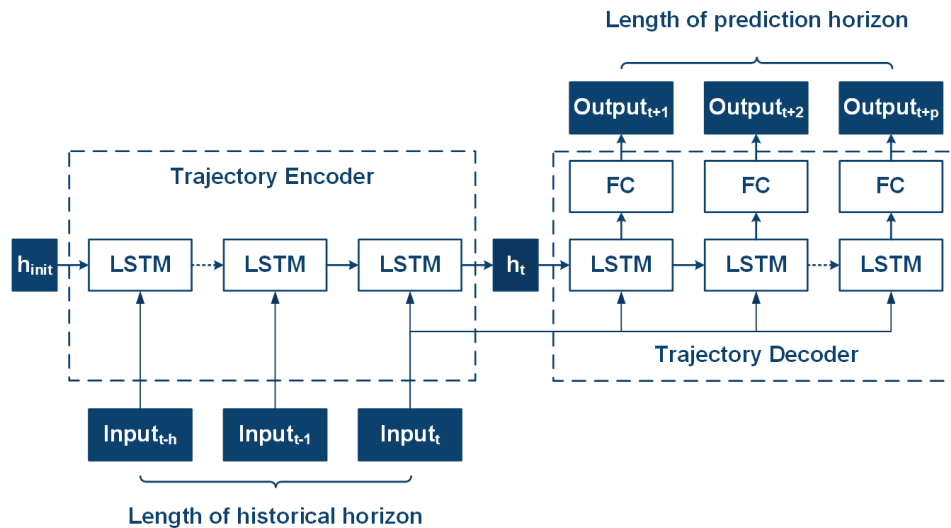


Figure 6-1 The structure of the proposed prediction model with the encoder-decoder structure. In this illustration, LSTM is used as an example in both trajectory encoder and decoder.

The details of the model are explained in the following sections. Both LSTM and GRU are tested in the prediction model to have a better understanding of their performance.

6.2.1 Trajectory Encoder

At the current time step, the trajectory encoder takes as input the status (positions and velocities) of the TA for the past h frames. For each past frame, the hidden state of the LSTM cell is updated by the corresponding TA status and the hidden state at the previous frame. The final hidden state of the LSTM cell in the trajectory encoder can be expected to encode the whole historical trajectory [60]. This context vector is then used as the initial hidden state of the decoder LSTM cell.

6.2.2 Trajectory Decoder

At each time step, for p frames into the future, the state of each LSTM cell in the trajectory decoder is updated based on the context vector from the encoder and agent status at the current frame. The context vector represents a summary of the agent's historical states. In the decoder, the input of each LSTM cell is the same and it is the TA status at the current frame. This residual connection between the current status and the trajectory decoder provides a reference for the initial status of the prediction, which in our case is the current position and velocity of the agent.

The final outputs of the trajectory decoder are the agent positions at each time step across the prediction horizon. A fully connected layer is used to match the dimensions of the output of each LSTM cell with the final prediction results.

6.3 Experimental Evaluation

Numerical experiments have been conducted on the inD dataset [20]. Details of the dataset used in the experiments can be found in section 4.4.5.

6.3.1 Compared Models

The purposes of model performance comparison are four folds:

1. Explore the influence of hidden state size on the model performance.
2. Study the influence of different RNN units on the prediction accuracy.
3. Study the influence of input of the decoder on the model performance.

4. Compare the prediction performance of DL-based methods and physical-based methods.

To achieve the above purposes, experiments are designed as follows:

1. The structure in Figure 6-1 is used and LSTM is selected. Hidden state sizes of 64, 128 and 256 are set to test the influence of hidden state size.
2. The structure in Figure 6-1 is used and LSTM and GRU are tested separately, with a fixed hidden size of 128.
3. The structure in Figure 6-1 is used and LSTM is selected. The inputs of the decoder are set to be zeros, position only, position and velocity.
4. The structure in Figure 6-1 is used and LSTM is selected. The Constant Velocity (CV) model and Constant Velocity and Turning Rate (CVTR) model are selected for comparison. The details of these two models are represented in Section 4.5.6.

6.3.2 Trajectory Prediction Results

6.3.2.1 The Influence of The Sizes of Hidden States

The ADE and FDE results are shown in Table 6-1. It can be seen that, with the increase of the LSTM hidden state size, the prediction errors decrease in both ADE and FDE. As the hidden state size increases from 64 to 256, there is a clear trend of improvement in all metrics. ADE decreases significantly from 0.6357 when the hidden state size is 64 to 0.3751 with a hidden state size of 256. FDE shows a similar trend, dropping from 1.7083 to 0.9058 as the hidden state size increases. RMSE also decreases from 1.350 to 0.472, which indicates a reduction in the average squared differences between the predicted and actual observations.

Table 6-1 The influence of the size of parameters

Hidden states size in LSTM	ADE	FDE	RMSE
64	0.6357	1.7083	1.350
128	0.5085	1.2706	0.821
256	0.3751	0.9085	0.472

6.3.2.2 The Influence of The Inputs to The Decoder

To eliminate the influence of parameter settings on prediction accuracy, A fixed hidden state size of 128 is used in the model and the LSTM is selected in the trajectory encoder and decoder. The prediction results are shown in Table 6-2.

The results indicate that when both position and velocity are used as LSTM inputs in the encoder and decoder (Test3), the prediction errors (both ADE and FDE) are substantially lower compared to the other two variations. In Test1, the inputs of LSTM in the decoder are zeros. However, the prediction model still outputs predictions with comparable accuracy, with an increase of 34.1% and 21.8% in ADE and FDE respectively. In Test2, with loss of velocity information, the prediction errors increase in both ADE (around 66.6%) and FDE (around 61.3%) compared to results with both position and velocity as inputs.

Table 6-2 The prediction performance with different LSTM inputs

	Encoder		Decoder		ADE	FDE	RMSE
	position	velocity	position	velocity			
Test1	√	√			0.6820	1.5476	
Test2	√		√		0.8470	2.0490	
Test3	√	√	√	√	0.5085	1.2706	

To have a better visualization of the displacement errors (DEs), the average DEs across all agents of a certain type and following a certain manoeuvre are plotted along the prediction horizon in Figure 6-2, Figure 6-3 and Figure 6-4. Generally, the supplement of velocity information in the trajectory encoder helps to improve the prediction accuracy of all three manoeuvres, by comparing DEs in Test 1 and Test 3 to those of Test 2. The velocity information helps to improve the going-through prediction accuracy of cars dramatically, as shown in Figure 6-2.

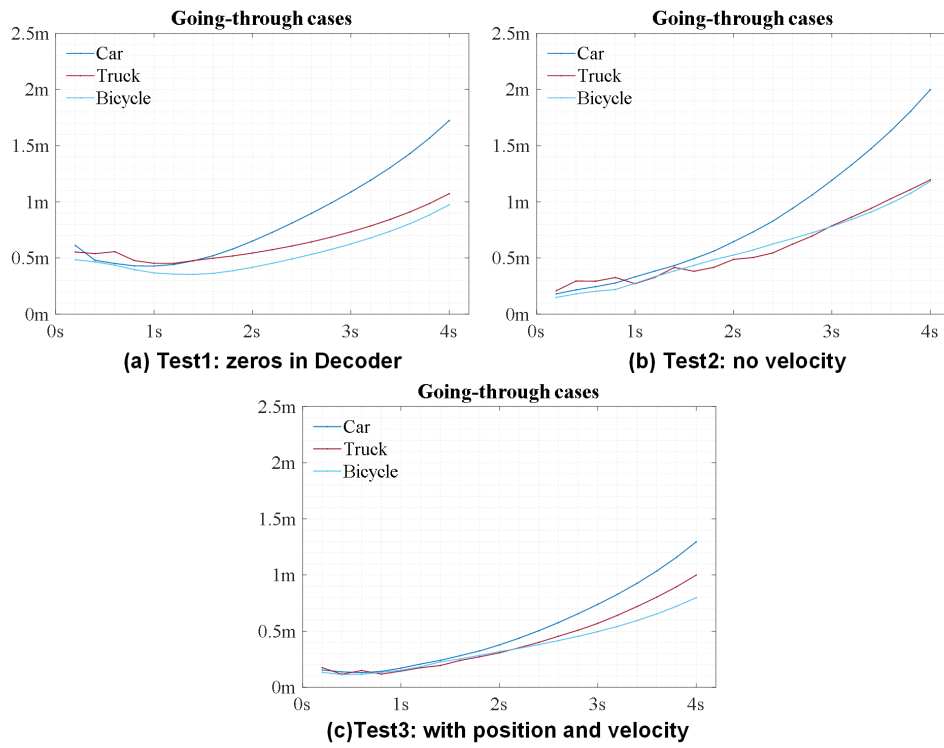


Figure 6-2 DEs in the going through case

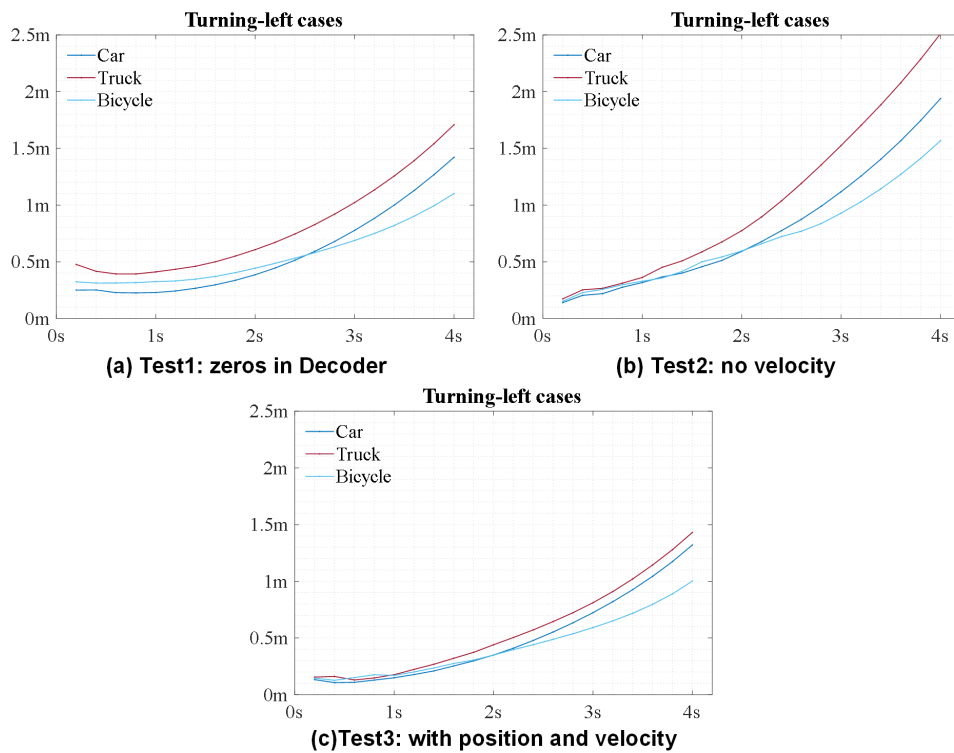


Figure 6-3 DEs in the turning-left case

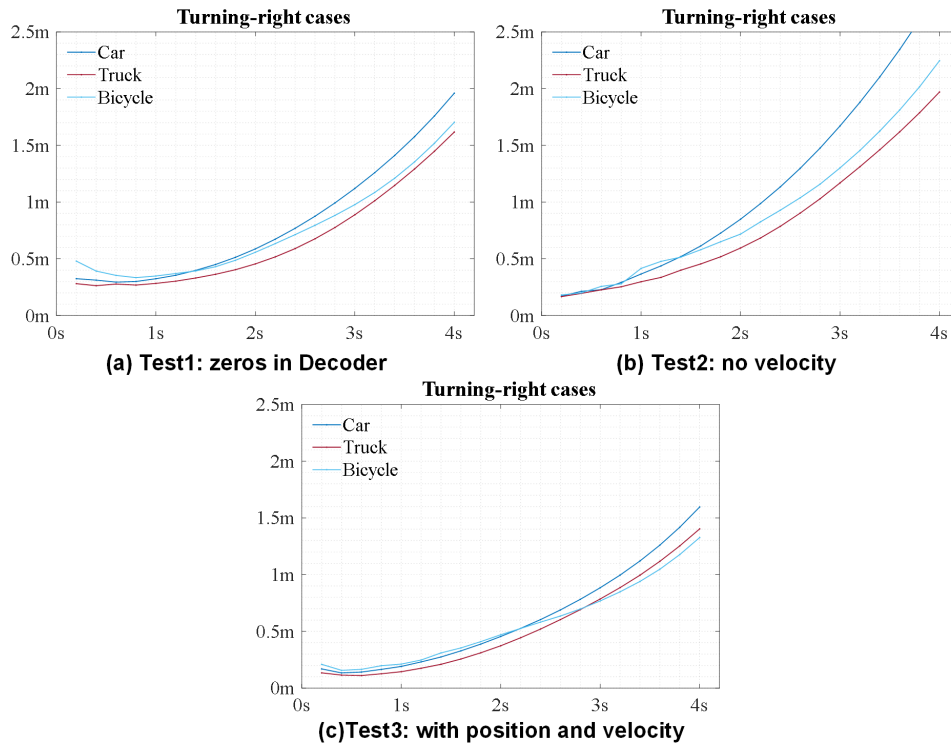


Figure 6-4 DEs in the turning-right case

Focusing on the first prediction step (at 0.2s), a residual connection between the current TA status and the decoder LSTM input helps to lower the prediction error, as shown in the comparison between Test2, Test3 and Test1.

6.3.2.3 The Performance of The LSTM and GRU

To compare the performance of different RNN units in the trajectory prediction task, LSTM and GRU are tested separately in the encoder-decoder structure. The hidden states of these two RNN variations are set to 128. The prediction results are shown in Table 6-3.

The LSTM model performs better across all metrics compared to the GRU model when the input of the decoder is the current status of the TA. When switching from LSTM to GRU, the ADE and FDE increase by 12.0% and 8.5% respectively. This suggests that LSTM may be more effective at handling long dependencies and maintaining memory information over sequences.

Table 6-3 The results of using LSTM and GRU

Enc	Dec	Dec Inp	ADE	FDE	RMSE
GRU	GRU	currState	0.5693	1.3888	0.9881
LSTM	LSTM	currState	0.5085	1.2706	0.821

6.3.2.4 Comparison to Physical-based Models

Physical-based models, such as the constant velocity model (CV) and constant velocity and turning rate model (CVTR), are easy and efficient to implement. However, they have limited performance in modelling the non-linear performance of agents' motion. To better illustrate their performance, the prediction results of the CV and CVTR models are compared with those of the LSTM. The LSTM model of Test3 in Table 6-2 is selected as the benchmark. Details of the used physical-based models are in Chapter 4. The comparison results are shown in Table 6-4.

Table 6-4 The FDE and ADE values by actor classes and manoeuvre classes

		CV		CVTR		LSTM	
		ADE	FDE	ADE	FDE	ADE	FDE
Overall		7.7997	15.841	5.0899	6.3347	0.5134	1.3083
Car	overall	7.3833	15.111	5.2207	6.6930	0.5301	1.3724
	gTh	13.918	27.466	10.087	8.9491	0.5163	1.2955
	tl	3.8977	8.3151	2.5661	4.5246	0.5003	1.3204
	tr	6.6683	14.239	4.8254	8.6849	0.6180	1.5964
Truck	overall	6.0940	12.879	4.5851	6.0935	0.5287	1.3565
	gTh	11.195	22.448	7.3490	5.5267	0.4120	0.9994
	tl	7.7673	17.018	5.1656	6.8885	0.5721	1.4310
	tr	3.8498	8.0933	3.5566	5.7792	0.5331	1.4035
Bicycle	overall	10.830	21.141	4.5734	4.4254	0.4140	0.9335
	gTh	13.503	26.248	5.2235	4.3154	0.3662	0.7985
	tl	5.6426	10.978	3.2525	4.1267	0.4329	1.0019
	tr	11.060	22.099	4.6025	5.4534	0.5643	1.3271

From the results, the trajectory prediction performance of LSTM surpasses those of CV and CVTR, both in ADE and FDE. The CV and CVTR achieve extremely large prediction errors in the going-through cases. Normally, the values of ADE are smaller than FDE in a trajectory prediction model. However, in the CVTR, going-through cases of all agent types have larger ADE than FDE.

Visualizations of predicted trajectories based on the CV model are shown in Figure 6-5. In (a), the traffic agents have a turning manoeuvre. From the visualization, the CV model does not have the ability to predict turning motions.

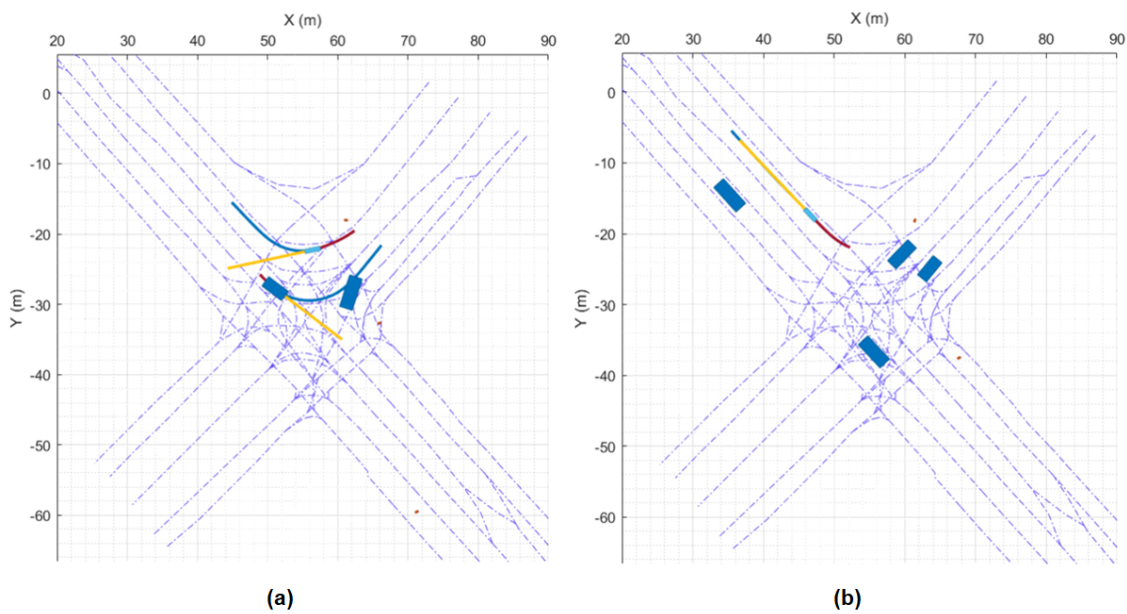


Figure 6-5 Visualization of the prediction results in the turning (a) and going through (b) cases. Blue lines are ground truth. Red lines are historical trajectories. Yellow lines are predicted trajectories.

The prediction errors of some typical motions are shown in Figure 6-6. Generally, the prediction model has smaller prediction errors in the going-through manoeuvres (0-5s in Figure 6-6) and larger errors in the turning manoeuvres (5 – 20s in Figure 6-6). These prediction errors grow through the prediction horizon.

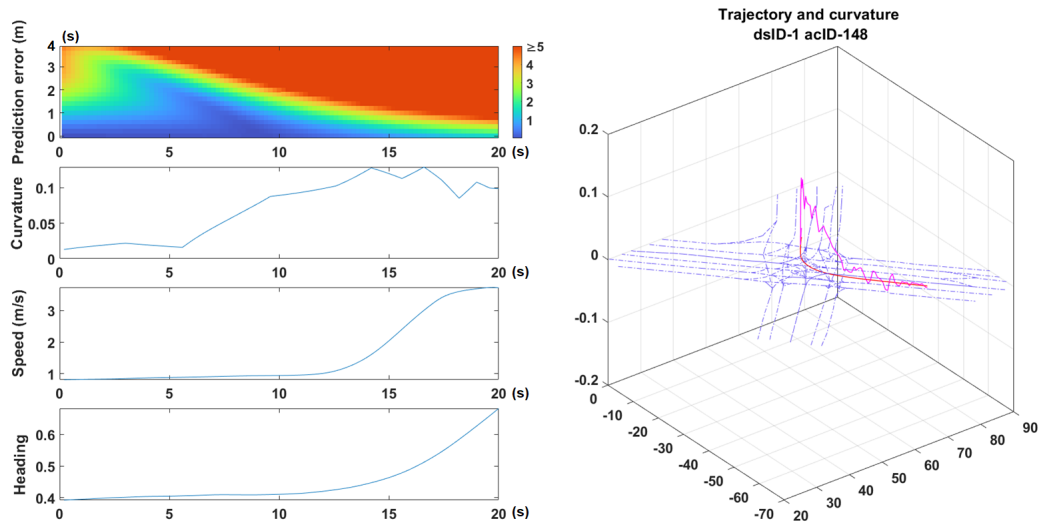


Figure 6-6 The prediction results of an agent (Dataset ID 1, Actor ID 148) are illustrated here. The agent moves on the road around 20s (the x-axis on the left side). In the prediction error, prediction errors across the prediction horizon (4s) are plotted as the heatmap at each time step. To better show the agent dynamics, trajectory curvature (rad), speed (m/s) and heading angles (rad) are also plotted. On the right side, the trajectory of the agent (in red) and the trajectory curvature (in pink) are plotted.

Visualizations of predicted trajectories based on the CVTR model are shown in Figure 6-7. Compared to the visualisation results of the CV model, the CVTR model has the ability to predict the turning manoeuvres.

The prediction errors of some typical motions are shown in Figure 6-8. From the prediction errors in Figure 6-8, although the model can make predictions on turning manoeuvres, the prediction error is still high. These prediction deviations are caused by underestimation of the length of the trajectories due to the constant velocity assumption. As shown in Figure 6-7, the predicted trajectories are shorter than the ground truths.

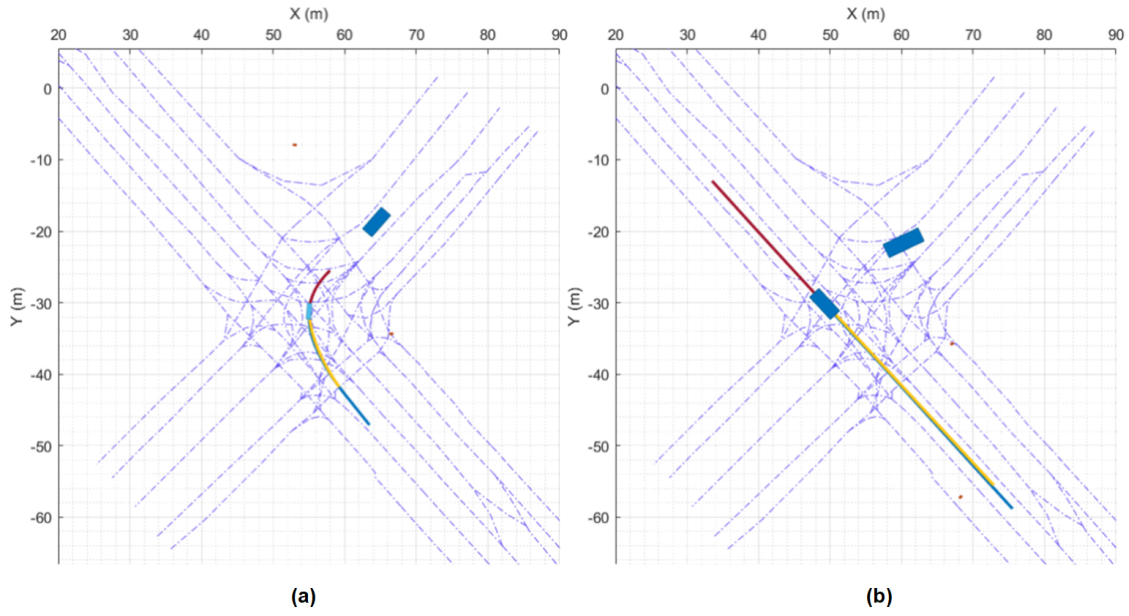


Figure 6-7 Visualization of the prediction results in the turning (a) and going through (b) cases. Blue lines are ground truth. Red lines are historical trajectories. Yellow lines are predicted trajectories.

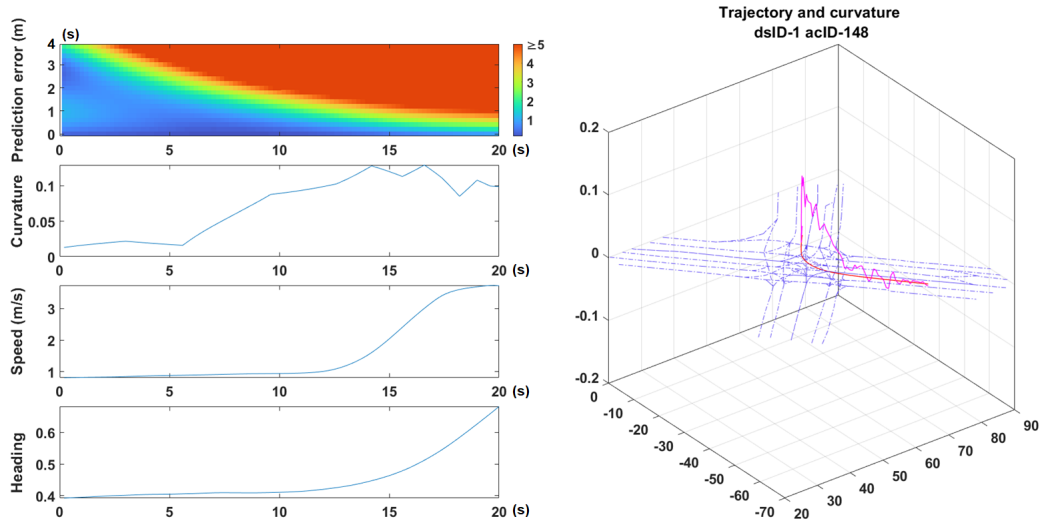


Figure 6-8 The prediction results of an agent (Dataset ID 1, Actor ID 148) are illustrated here. The agent moves on the road around 20s (the x-axis on the left side). In the prediction error, prediction errors across the prediction horizon (4s) are plotted as the heatmap at each time step. To better show the agent dynamics, trajectory curvature (rad), speed (m/s) and heading angles (rad) are also plotted. On the right side, the trajectory of the agent (in red) and the trajectory curvature (in pink) are plotted.

6.4 Conclusions and Future Work

This chapter presents a trajectory prediction architecture based on the RNNs and an encoder-decoder structure. The comparisons of different hidden state sizes, input data types and variations of LSTM and GRU have shown that:

1. With sufficient training data, the increase in the hidden state sizes of the LSTM helps to improve the prediction accuracy. Larger hidden state sizes in LSTM models are generally associated with more effective learning of dependencies in the data, leading to more accurate predictions.
2. Velocity information as additional inputs helps to improve the prediction accuracy, especially in the going-through manoeuvres. And residual connection from the current input to the LSTM input in the decoder helps to lower the prediction error at the initial prediction step. The model's ability to utilise both position and velocity inputs likely helps it capture more dynamic features of TAs' movement, leading to more accurate predictions.
3. Trajectory prediction could be operated purely based on the output of the trajectory encoder, with zeros as input of the trajectory decoder. This highlights the importance of the historical information extracted by the RNN encoder in the prediction task.
4. Compared to CV and CVTR models, the model based on LSTM and the encoder-decoder structure has much better performance considering the ADE and FDE.

The research in this chapter benefits the following research work by giving an insight into choosing suitable architectures, RNN units and hyperparameters, like hidden state sizes.

For a better prediction performance, it is worth exploring the influence of neighbour interactions and road layouts on the trajectory prediction.

7 Trajectory Prediction by Considering The Interactions between The Traffic Agents

7.1 Introduction

Traffic participants are continually engaged in an interactive environment, in which their movement is a complex interplay of two key factors: adherence to their historical patterns and the influence of surrounding agents (SAs). This dual influence often results in non-linear manoeuvres of drivers.

The need for participants to negotiate with each other is paramount. Understanding the motivations of other traffic agents and adjusting movements in response to these dynamics is crucial for maintaining safety and enhancing traffic flow efficiency. These interactions are not only common but also highly variable and particularly intense in urban environments, where the density of traffic and diversity of TAs, including vehicles, pedestrians, and cyclists, add complexity to interactions.

Since the interactions are common and have an influence on the TAs' moving patterns, it is necessary to consider interaction in the trajectory prediction.

7.1.1 Related Research

SAs are usually identified if they have a potential influence on the motion of the TA. The definition of SAs and their identification methods can be found in section 3.5.2, as well as the commonly used methods to extract neighbour interaction information.

7.1.2 Motivations and Contributions

This work serves as a foundational step toward integrating various sources of information, such as historical states, road layouts and neighbours, into the prediction model. The goal is to develop a fusion framework that combines these inputs for enhanced trajectory prediction. Specifically, this study applies a prediction model derived from the literature [1] and incorporates the SAs' information into the backbone model, in which SAs are arranged in a fixed-size

grid and the Convolutional Social Pooling (CSP) method is used to extract agent interactions.

Adaptation is made to the model inputs for this project's purpose and the influence of SAs on trajectory prediction accuracy is studied. To investigate the appropriate model structure for the integration of the agent interaction encoding and the agent's status data, variation is made on the derived model and the performance of these two models is compared to study their performance. The compared models are named LSTM_CSP1 and LSTM_CSP2 separately. In both LSTM_CSP1 and LSTM_CSP, the input of the trajectory decoder is the concatenation of agent historical trajectory encoding and neighbour interaction encoding. The difference is that, in LSTM_CSP1, the hidden states of the LSTM in the trajectory decoder are initialised by zeros while they are initialised by agent historical trajectory encoding in the LSTM_CSP2.

The framework developed in this study will be further used in Chapter 8, where road layout information will be integrated into the framework.

This work is significant to the trajectory prediction communities because it provides valuable data on the influence of neighbours on the prediction accuracy. The modified structure for integrating agent interaction encodings and the agent's status data improves the trajectory prediction accuracy. And the exploration of different integration methods guides the design of the model architecture

In the rest of this chapter, Section 7.2 presents the trajectory prediction model. Experimental evaluation is performed in Section 7.3. Finally, the conclusions and discussions are in Section 7.4.

7.2 Prediction Model

The aim is to predict future positions for TAs, which is a regression task. An encoder-decoder prediction structure is used. The structure contains a trajectory encoder shared with the TA and SAs. Then a Convolutional Social Pooling (CSP) [1] is used to extract the agent interactions, producing a social context encoding. The encoding of the TA's historical trajectory and that of the social context are put into the trajectory decoder for final prediction.

In an LSTM-based decoder, there are two ways to input the TA's trajectory and social context encoding. The first approach is to concatenate the social context encoding with the TA's trajectory encoding and use this concatenation as the input of the LSTM cell in the decoder. In this case, the hidden states of the LSTM are initialised by zeros. This model structure is derived from [1], which proposes CSP. The original model concatenates the probabilities of manoeuvres, right turning, accelerating etc., with the social context encoding to generate the predictive distribution for future motion. This deviates from the regression target in this research project and for adaption, manoeuvre probalistics are replaced by the trajectory encoding of the TA. This derived structure is denoted as LSTM_CSP1 in the later sections and its details are shown in Figure 7-1.

The second approach is to feed the trajectory encoding of the TA to the LSTM hidden states as an initialisation. In contrast, the social context encoding is fed into the LSTM input, with a concatenation of the TA's current status. This structure will be denoted as LSTM_CSP2 in the following and is shown in Figure 7-2.

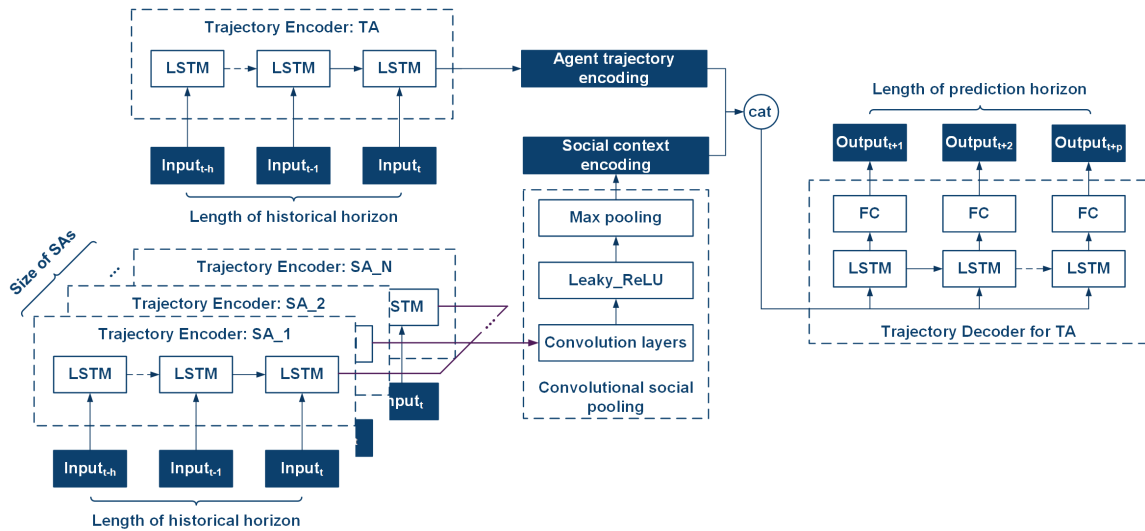


Figure 7-1 The model (LSTM_CSP1) contains a shared trajectory encoder for TA and SAs, a Convolutional social pooling layer and a trajectory decoder. The input of the LSTM is the concatenation of the agent trajectory encoding and social context encoding. The hidden states of the LSTM will be initialised by zeros in the decoder.

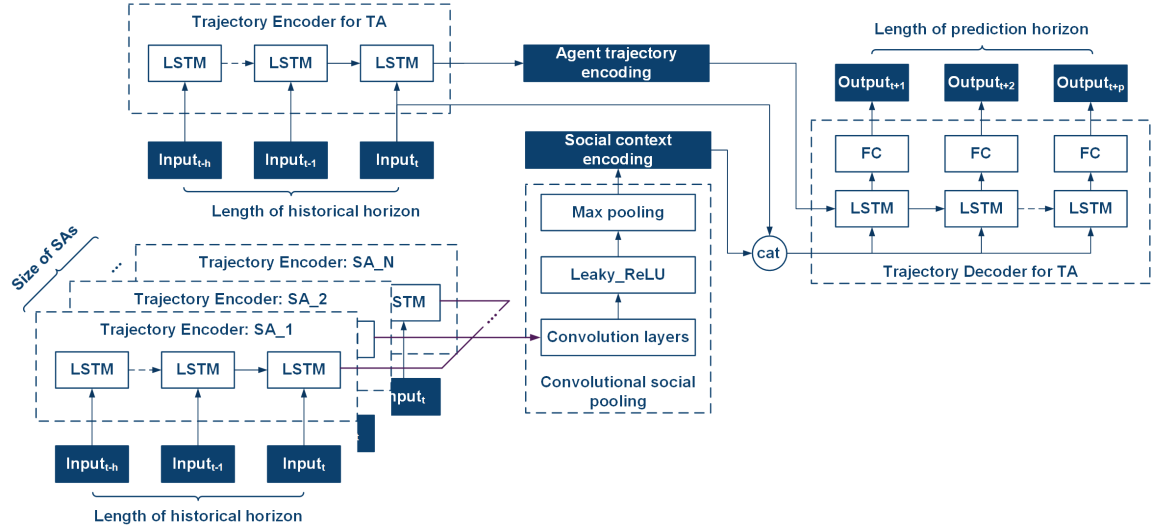


Figure 7-2 In this structure (LSTM_CSP2), the input of the LSTM is the concatenation of social context encoding and TA's current status. The hidden states of the LSTM will be initialised by the trajectory encoding of the TA.

7.2.1 Trajectory Encoder

An LSTM trajectory encoder is used for learning the dynamics of agent motion. For each instant, snippets of the most recent h frames of trajectory history are passed through the LSTM trajectory encoder. The same LSTM trajectory encoder is used for the TA and all SAs. The final LSTM state -- the final hidden state of the LSTM -- for each agent can be expected to encode its historical motion states.

7.2.2 Convolutional Social Pooling

The LSTM trajectory encoder focuses on the historical motion features of individual agents. Without any inputs describing agent interdependencies in the traffic scene, its resulting trajectory encoding fails to capture agent interaction context. To remedy the deficiency of lacking agent interaction input, [88] proposed social pooling to enable spatially proximal traffic agents to share information with the TA. This method first defines a spatial grid around the TA and populates each grid cell with the output of the agent trajectory encoder, if the agent is inside the grid cell. This populated grid is denoted as a social tensor and an example can be found in Figure 7-3. This social tensor can be flattened by

concatenating all grid encodings end to end and be used as an additional input to the model, in addition to the LSTM state of the TA. Experiments have shown that the social tensor can improve the trajectory prediction accuracy [64] [1].

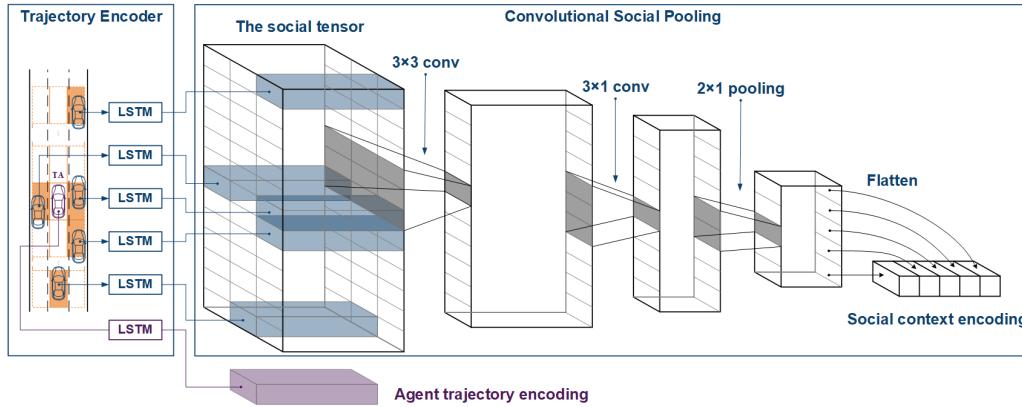


Figure 7-3 Demonstration for the module of CSP. The trajectory encodings of SAs are filled into the grid to form the social tensor. Then two convolutional layers and one max-pooling layer are used to extract the social context encoding.

However, this simple concatenation causes a loss of spatial characteristics. This deficiency is further addressed by extracting interaction context through convolutional and pooling layers, denoted as convolutional social pooling [1]. This structure preserves the spatial structure of the social tensor, compared to the deployment of a fully connected layer or simple concatenation [88] [64]. The convolutional layer distinguishes the cells adjacent to the TA and those far away. It can be expected to learn useful local features within the receptive field of a convolutional kernel, a group of grid cells. The following max-pooling layer helps to improve local translational invariance which increases the robustness in case of an inaccurate assignment of SAs to one of the grid cells.

In the application, the grid size centred at the TA is 13 by 3 in the longitudinal and lateral direction of the TA's motion separately. Each grid cell has a length of 5m and a width of 3.5m. The first convolution layer has a kernel of 3 by 3 with an input channel dimension equal to the dimension of LSTM hidden state size in the trajectory encoder. The output channel is set to be 64, and this is also the input channel of the next convolutional layer. The second convolutional layer has a kernel of 3 by 1 with its output channel equal to 16. At last, there is a max pooling

layer with a kernel size 2 by 1. The final output dimension of convolution social pooling is 80, which is also called social context encoding.

7.2.3 Trajectory Decoder

The input of each LSTM in the trajectory decoder is the concatenation of the TA's trajectory encoding through the trajectory encoder and the social context encoding through the CPS layer. At each time step, for p frames into the future, the state of each LSTM cell in the trajectory decoder is updated based on the concatenated encoding and the LSTM hidden state at the previous instant. The decoder outputs at each time step future locations of the TA at that time instant, with a fully connected layer used to match the dimensions of the output of each LSTM cell with the final prediction results, the trajectory points.

In the LSTM_CSP1, the LSTM inputs in the decoder will be the concatenation of the TA's trajectory encoding and social context encoding. LSTM's initial hidden states are set to be zeros.

In the LSTM_CSP2, the concatenation of social context encoding and TA's current status will be used as LSTM inputs. LSTM hidden states will be initialised by the TA's trajectory encoding.

7.3 Experimental Evaluation

Numerical experiments have been conducted on the inD dataset [20].

7.3.1 Compared Models

The purpose of model performance comparison is to validate if the CSP and the corresponding structure could improve the prediction accuracy. Thus, the three models are compared:

- Reference designs of LSTM_CPS1 and LSTM_CSP2,
- An LSTM-based encoder-decoder prediction model based on the historical status of TA.

7.3.2 Trajectory Prediction Results

7.3.2.1 The ADE and FDE

The prediction performance of the three models is listed in Table 7-1. Across all classes and manoeuvres, the LSTM_CSP1 generally has higher ADE and FDE values compared to the standard LSTM and LSTM_CSP2. This suggests that the structure of LSTM_CSP1 is less accurate in the prediction work. For all agent classes, the predictions on the going-through cases have the biggest errors compared to the other two manoeuvres in the LSTM_CSP1. However, going-through cases generally have the highest prediction accuracy without CSP.

The LSTM_CSP2 shows a much better performance than LSTM_CSP1 in ADE and FDE across all agent classes and manoeuvres. It achieves a better performance than the prediction model without CSP in the manoeuvre of left turning of all agent classes. It makes sense, because SAs have a bigger impact in left turning manoeuvres in a German dataset, given that left turns often require crossing of the opposite lane, and judging the incoming traffic.

Table 7-1 The FDE and ADE values by actor classes and manoeuvre classes

		LSTM_CSP1		LSTM_CSP2		LSTM	
		ADE	FDE	ADE	FDE	ADE	FDE
Overall		2.1377	3.1567	0.5026	1.2570	0.5134	1.3083
Car	overall	2.1803	3.2733	0.5214	1.3279	0.5301	1.3724
	gTh	2.9878	4.5064	0.5328	1.3013	0.5163	1.2955
	tl	1.7400	2.4156	0.4655	1.2159	0.5003	1.3204
	tr	2.1140	3.6026	0.6355	1.6239	0.6180	1.5964
Truck	overall	1.9207	2.9256	0.5075	1.2544	0.5287	1.3565
	gTh	2.7302	3.3807	0.4218	0.9392	0.4120	0.9994
	tl	1.6110	2.1459	0.5462	1.2944	0.5721	1.4310
	tr	1.8951	3.2593	0.5067	1.3106	0.5331	1.4035
Bicycle	overall	1.9947	2.6105	0.3968	0.8658	0.4140	0.9335
	gTh	2.2578	2.9025	0.3475	0.7372	0.3662	0.7985
	tl	1.4772	1.8355	0.3972	0.8601	0.4329	1.0019
	tr	2.0314	3.0712	0.5918	1.3889	0.5643	1.3271

7.3.2.2 The Displacement Error Visualization Across Prediction Horizon

The displacement errors (DEs) across the whole prediction horizon are shown in Figure 7-4. Generally, the prediction errors grow as the prediction horizon increases in both LSTM_CSP1 and LSTM_CSP2.

LSTM_CSP1 generally exhibits significantly higher Displacement Errors (DEs) compared to LSTM_CSP2, particularly in the 'going through' manoeuvre. In this scenario, variation in performance among the different agent types—cars, trucks, and bicycles—is large, with cars experiencing the largest prediction errors at the final time step. Across all three manoeuvres, a noticeable surge occurs at around 0.4 seconds in LSTM_CSP1, suggesting an inadequate initial estimation of the TA's status.

Conversely, LSTM_CSP2 maintains relatively low prediction errors at the initial stages of forecasting. This model's accuracy is consistently higher, showing a better grasp of the initial conditions across all agent types, which translates into more uniform prediction performance.

Additionally, both models exhibit more accurate performance in scenarios involving left turns as opposed to right turns. However, this trend is more distinct in LSTM_CSP1, where the difference between the manoeuvres is more pronounced. Despite these variations, LSTM_CSP2 consistently outperforms LSTM_CSP1, particularly in handling initial prediction challenges and achieving lower overall prediction errors.

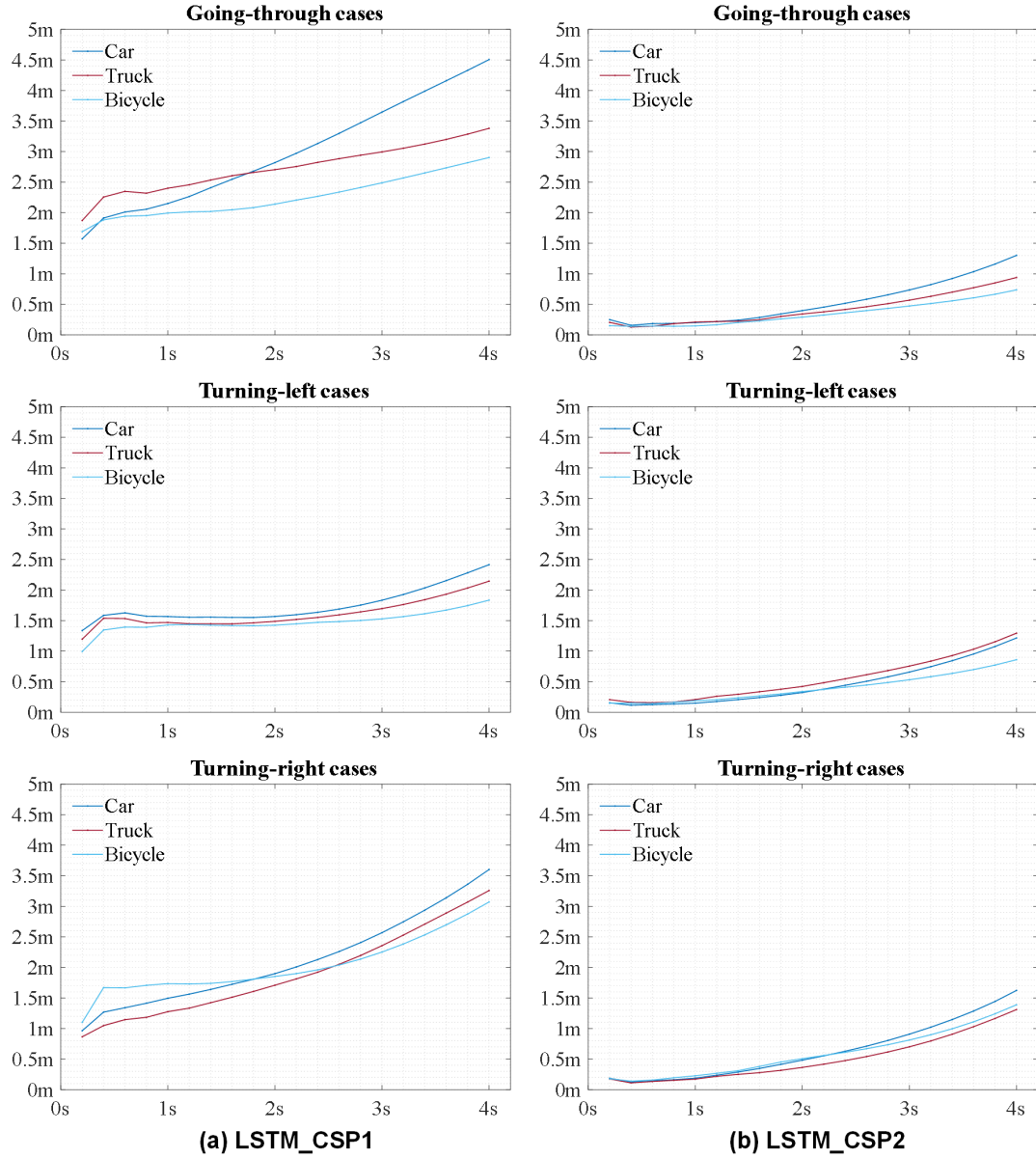


Figure 7-4 DEs across the prediction horizon for LSTM_CSP1 and LSTM_CSP2.

7.4 Conclusions and Future Work

This chapter presents a trajectory prediction architecture of TAs organised around an LSTM-based encoder-decoder structure and a Convolutional Social Pooling mechanism to mine the agent interactions. The comparisons of testing results have shown that:

1. Agent interaction mining has the potential to improve prediction accuracy, especially in left-turning cases.

2. While using an LSTM-based decoder, the initialisation of its hidden states is important. The TA's trajectory encoding works better than just zeros in the initialisation stage of LSTM hidden states.
3. The residual connection from the agents' current status to the decoder LSTM input improves the prediction accuracy at the initial prediction stage.

The research in this chapter will benefit the following research work by giving insight into choosing suitable architectures while coupling additional traffic context encodings into the encoder-decoder structure.

Though agent interaction encodings have improved the prediction accuracy, a more suitable neighbour identification architecture in the urban scenario and a more powerful interaction mining network are needed to maximise the benefits of including agent interaction information in the prediction model.

8 Trajectory Prediction With Road Scene Mining

8.1 Introduction

Map information, including road geometries and traffic rules, is essential in guiding traffic agents moving on the road. For example, the vehicles will follow the geometry of driving lanes and manoeuvre turning according to the lane connectivity. Dynamic relations between vehicles and road lanes, for example, the deviations from vehicles to lane centrelines, explicitly show drivers' intentions like lane keeping and lane changing. Experiments have shown that using additional map information can achieve better trajectory prediction performance [178].

Road layout (RL) maps are usually stored in the form of an RGB image or a directed graph in deep learning (DL) applications. Neural Networks (NNs) like Convolutional Neural Networks (CNNs), Graph Neural Networks (GNN) and Attention Mechanism (AM) are commonly applied in map feature extraction. The extracted high-dimensional map feature encodings are concatenated with those of the traffic agents to generate a map-aware input for the trajectory prediction decoder, which outputs the final predicted trajectories [117] [108].

8.1.1 Related Work

In the early stages, one common way to contain road geometry information in the prediction model is to represent the traffic agent status relative to the road elements. For example, in [57], the relative displacements to the lane markings and lane centrelines, as well as their first and second-order derivatives, are used as inputs of an LSTM model.

Another informative method to represent road scene contexts, including road geometries and traffic dynamics, is to form an image-like representation of traffic scenarios by discretisation technologies and then road scene features are extracted by CNN-based methods. Road scene discretisation is usually performed around the target agent (TA) in Bird-eye-view (BEV). In [96] [99], rasterisation is used to create RGB images, where distinct RGB colours are used to represent different static contexts of the vector RL map. These RGB images

are usually multiple-layer and stacked to include different road scene contexts, like driving lane polygons, road boundary polylines and directional lane centreline polylines. This stacked representation sometimes causes some information loss because the subsequent images will overlay the content of the previous one. A similar technology is also used in [83]. Different from overwriting subsequent images, a multi-channel image is created to store the rasterized road scenes without loss of spatial features. For example, the completeness of the lane markers will not be affected when they are obstructed by agent boxes since these two features are stored in different channels. Therefore, predefining orderings while stacking RGB images could be avoided. The main difference between multi-channel grid maps and RGB images is that the grid cells in grid maps are populated with numerical values instead of RGB values. This mechanism allows grid cells to store dynamic traffic contexts like traffic light states, which is not easy for RGB representation. To achieve a receptive-field-variable representation of the grid maps, single-channel grid maps with three different discretization resolutions are created to represent the scene. The three grid sizes are classified into 'local', 'medium' and 'global' to describe the scenario from micro trajectory features to the whole scenario context.

Some limitations lie in scene context representation and real-time application with image-like representation. First, stacking multiple layers with overwriting mechanisms inevitably results in information loss such as connectivity among lanes. Second, 2D convolution may not be able to capture connectivity between road lanes, especially when the receptive field of the convolution kernel is limited. For instance, a lane of interest may extend over a long distance along a particular direction. To capture this information, the receptive field should be large enough to cover not only the intended area but also extensive regions outside the lane. Additionally, adjacent lane pairs in the same or opposite directions have completely different semantic meanings and dependencies, which might not be adequately captured by convolution operations [105]. Additionally, this rasterization poses a challenge for real-time applications [54] because scenario rasterization should be operated within a specific domain around the traffic agent at each time step.

Apart from rasterized images, directed graphs are often applied to represent the traffic scenario. This type of representation relies on vectorized road-layout maps that summarise the geographic and semantic information of lane segments and their internal connectivity.

Basic graph components are nodes and edges, corresponding to the lane segments and their connectivity respectively. A lane segment is usually represented by a sequence of ordered lane centrepoints [102]. And the physical characteristics of these centrepoints are often used as the lane node features like their coordinates and orientations (e.g., yaw or tangent direction), which capture both the direction of the traffic flow and the legal routes that a driver can follow [103] [104] [105]. With those maps described by lane boundaries, a node could be represented by boundary points. For example, a graph node is represented by the start and end points of the boundary polyline in [101].

Individual lane nodes are linked by edges in the graph, each of which represents the connection of two separate lane segments in the RL maps [107]. According to the adjacency of the lane segments in the map, edges have types of predecessors, successors, and left and right neighbours. Given a lane segment, its predecessor and successor are other lane segments that can directly travel to and from the selected segment. Left and right neighbours refer to the lane segments which can be directly reached without violating traffic rules in the traversal directions [105].

The feature of a lane node defined above only captures the local information of a road layout map. To aggregate the topology information of the map graph at a larger scale, the encoded lane nodes exchange information by message propagation through the graph, updating lane node encodings by collecting encoding info from its surroundings. In some applications, message propagation among lane nodes is processed by pooling methods. In [104] [101], max-pooling is used to integrate neighbour node encodings. However, pooling methods only perform a one-step aggregation from a node's direct neighbours. To enable a multi-step aggregation which will reach indirect neighbours with an exploration depth in the graph topology, GCN could be used [86]. In [105], a graph

convolution method is used to realize multi-scale message propagation along different directions of the lane nodes. Because traffic agents usually move farther in the lane longitudinal direction than the lateral direction over the prediction horizon, the graph convolution operates within four separate adjacency matrices to enable exploration in predecessors, successors, left neighbours, and right neighbours with diverse depths. To accommodate the long-range dependency of lane node information, the graph convolution is further set to be expanded to enable exploration with customized depth along the lane edges, mimicking the long-range exploration ability of the dilated 2D convolution operator by enlarging the receptive field. In [104], GAT is used to realize message propagation according to the importance of the neighbours. The attention weights are calculated based on encodings of node features and the node aggregates neighbour encodings through weighted sum.

The issue of limited receptive fields in the combination of rasterized images and CNNs could be solved by multiple-step feature aggregation. For efficiency, graph layers of macro levels could be used to explore global topology, in which lane segments have a length of 10 to 20m compared to micro levels with a length of 1m [86] [104]. This macro-level graph is combined with micro-level ones to represent the local scenario context while being aware of global topology. However, a fine-grained graph with high lane resolution usually challenges the on-chip memories and real-time computation. Another issue of graph-based representation is that it is hard to relate one traffic agent to one specific lane node, especially at the intersection where some lane nodes have space overlapping.

Considering these limitations, AM methods are promising because lane nodes could be selected independent of the node's internal connectivity and the vehicle-lane matching could be performed without geometry overlapping. AM methods try to guide the target vehicle towards the most likely lane segments in the future. In [178], the deviations from the vehicle to its surrounding lanes are calculated across the historical horizon by calculating the difference between the vehicle's locations to their projections on each lane centreline. These lane-vehicle pairwise deviations are further passed to the linear projection layer to calculate the

attention scores which indicate the possibility of the vehicle referring to the lane. A comprehensive encoding is then obtained by the weighted sum of relevant lanes' feature encodings and their attention scores. In [112], lane piece features are extracted by deploying 1D convolution (1D Conv) and MLP on the lane centre point sequence. Possible lanes within a certain distance surrounding the vehicle are extracted and then the dot-product attention module is applied to predict the probability of each lane that the vehicle is heading towards. More specifically, the vehicle trajectory features are dot-product with the lane features to obtain the probabilities associated with each lane. To narrow down the attention scope, in [117], lane segments within 50m of the vehicle are isolated from the map and 3 most likely lane segments that the vehicle might refer to are selected for encoding concatenation. This is realised by cross attention between lane and vehicle encodings. In detail, the encodings of lanes and vehicles are treated as key-value pairs and query vectors separately. Then scaled dot-product is used to calculate the probability scores of each lane segment.

For AM methods, a larger number of target lane segments increases the chances of containing irrelevant lane segments, which might confuse the model in training. On the other hand, too small a number might cause insufficient searching in the local scope, especially when the vehicle drives on the reverse lane segments temporally, to avoid obstacles. A range search or a nearest neighbour search in lane segment selection tends to include more lane segments in the lateral direction while fewer in the longitudinal. This might limit the AM's ability to identify the potential relevant lane segments because vehicles might have longer travel distances in the longitudinal direction than the lateral.

8.1.2 Motivations and Contributions

This chapter aims to propose a solution to use map information to improve the accuracy of trajectory prediction. A sparse road layout map is used as a supplement to provide traffic scene context. To validate the benefits of introducing map information, the prediction model is trained and tested in three urban scenarios.

The novel contributions of this chapter are:

1. Instead of fine-grained map elements, sparse RL maps are used in the trajectory prediction model for performance enhancement.
2. Based on the selected map elements, agent-map attention is used to extract comprehensive map encodings to enable map-aware trajectory prediction.

This work is significant to the trajectory prediction research because:

1. This work visualises the learned attention scores and provides intuitive explanations of the model behaviours. This good interpretability can also benefit other downstream modules of an autonomous driving system.
2. The application of sparse RL maps could handle multiple scenario information in one model which benefits the model training among multiple scenarios with tight storage space.
3. The study on map-enhanced trajectory prediction shows the benefits of using sparse RL maps for prediction performance enhancement.

8.2 Prediction Model

The aim of this research is to predict future positions for the TA across a prediction horizon, which intrinsically is a regression problem. The framework studied in Chapter 7 is used to integrate the road layout information. The proposed model is shown in Figure 8-1.

8.2.1 Map Element Feature Encoder

The map elements are represented by a bunch of sequential centrepoints, starting from its entrance to the exit. A group of 1D convolution layers and max-pooling layers is used to produce an implicit representation of the map element features, targeting encoding geometry characteristics like the shape, direction and curvature.

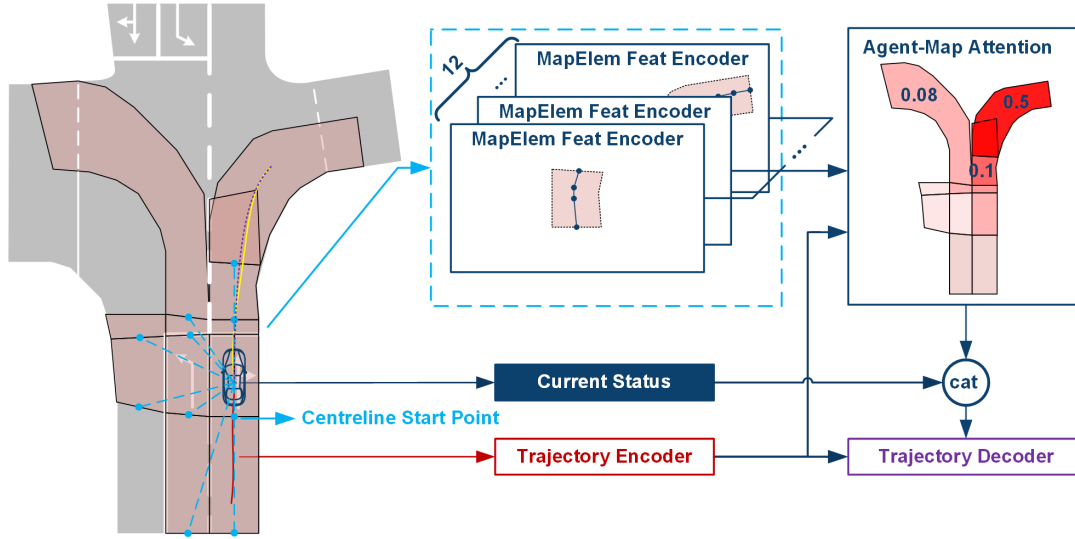


Figure 8-1 The grey background is the intersection surface and white lines are the lane markings. The red line is the historical trajectory of the TA. The yellow line and purple dotted line are the ground truth and predicted trajectory in the prediction horizon. 12 map elements are selected according the distance from the TA to the start point of the centreline and they are marked with pink polygons. For conciseness, centrelines are not visualised. Map element feature encoders are applied to process the centre points of map elements. The resulting map element feature encodings are imported into the agent-map attention module, along with the encoding of the historical trajectory, to produce the agent-map attention encoding. This agent-map encoding will be concatenated with the current status of the TA and serve as inputs of the trajectory decoder to produce the final prediction.

The input is the map element feature matrix, of dimension $m \times D_{pos} \times n_{ct}$, where m corresponds to the number of map elements, D_{pos} to the size of the centrepoint feature (x and y coordinates), and n_{ct} to the number of centrepoints.

There are three layers of 1D convolution with a kernel size of 3, and two max-pooling layers in between them with a kernel size of 2. No padding is used in the convolution layers. Through the map element feature encoder, the representation of each map element is converted from a sequence of centrepoints to an array of high-dimension features.

These encodings will be further processed with other encodings, like the encoding of the TA's historical states, and a linear projection layer is used to match the size of map encodings to that of the others.

8.2.2 Trajectory Encoder

The motion status of each TA is encoded by using an LSTM encoder. At any time t , a sequence of h time steps of the trajectory history is fed into the encoder. The LSTM states in the encoder are updated recursively over the h time steps. The final LSTM state can be expected to encode the motion status of that TA. The sequential inputs are:

$$X_{status} = [X_{status}^{(t-h)}, \dots, X_{status}^{(t-1)}, X_{status}^{(t)}] \quad (8-1)$$

where h is a fixed time horizon, and at any time instant t ,

$$X_{status}^{(t)} = [p_x^{(t)}, p_y^{(t)}, v_x^{(t)}, v_y^{(t)}] \quad (8-2)$$

8.2.3 Agent to Map Element Attention

Cross-attention is used in this research to map the TA's historical encoding and map element feature encodings to get the agent-map attention context. This attention context contains map geometry information which the TA uses to make decisions while moving.

In practice, the query vector is represented by the TA's historical encoding and key-value pairs are from the map element feature encodings. The actor-map attention context is represented by:

$$H_{ac2map} = \text{Attention}(Q_{actor}, K_{map}, V_{map}) = \text{Softmax}(Q_{actor} K_{map}^T) V_{map}, \quad (8-3)$$

where H_{ac2map} is the actor-map attention context in batches.

$$Q_{attn} = H_{actor} W^K = [h_{actor}^1; \dots; h_{actor}^{n-1}; h_{actor}^n] W^K, \quad (8-4)$$

$$K_{attn} = H_{mapElem} W^K = [h_{mapElem}^1, \dots, h_{mapElem}^{m-1}, h_{mapElem}^m] W^K, \quad (8-5)$$

$$V_{attn} = H_{mapElem} W^V = [h_{mapElem}^1, \dots, h_{mapElem}^{m-1}, h_{mapElem}^m] W^V, \quad (8-6)$$

where $h_{mapElem}^k$ is the k_{th} map element feature encoding and m is the number of map elements, W^k and W^V are learnable parameters to convert map element feature encodings into key and value vectors. where h_{actor}^k is encoding of k_{th} agent's historical status and n is the batch size, W^Q is the learnable parameter performing linear projection on actor historical status encodings for the purpose of getting query vectors.

In this research, map elements from all scenarios are assembled in the $H_{mapElem}$. With no distinction of map elements from different scenarios, the TA tends to refer to every map element in the container to calculate the corresponding weight values. However, the TA should only focus on its current scenario to avoid any confusion for DL-based models. To avoid this issue, a 3D attention mask is created to prevent attention to certain map elements. This mask is of shape $(N_{batch}, L_{attnTgt}, m)$ and allows for a separate mask for each map element in the training batch, where N_{batch} corresponds to the batch size and $L_{attnTgt}$ is the output length of the agent-to-map attention. In this research, $L_{attnTgt} = 1$.

In the created 3D attention mask, a value of 1 indicates that the corresponding map element is not allowed to attend.

8.2.3.1 Trajectory Decoder

An LSTM decoder is used to generate the future trajectories of the TA [92]. At any time t , the decoder generates the future trajectories over the future p time steps at once. At each time step, the decoder outputs a deterministic coordinate:

$$Y^{(t)} = [P_x^{(t)}, P_y^{(t)}] \quad (8-7)$$

In the decoder, the input of the LSTM is the concatenation of the current status of the TA and the agent-map attention context. The hidden states of the LSTM in the decoder are initialised by the output of the trajectory encoder.

8.3 Experimental Evaluation

Numerical experiments have been conducted on the inD dataset [20].

8.3.1 Compared Models

For comparison purposes, the following variations of the reference design are tested:

- Backbone LSTM model taking historical status as inputs,
- LSTM model with convolutional social pooling encoding neighbour interaction,
- LSTM model with agent-map attention

8.3.2 Implementation Details

The hidden state sizes of the LSTMs in both the trajectory encoder and decoder are set to 128. The vector dimension sizes in the agent-map attention are also set to 128. The output channel dimensions for the three ID Conv layers are set to 64, 64 and 128 respectively.

8.3.3 Trajectory Prediction Results

8.3.3.1 Overall Performance of Prediction Models

The comparison of the three models is shown in Table 8-1. Generally, the LSTM_MAP and LSTM_CSP achieve better prediction performance than the backbone LSTM.

The LSTM_MAP achieves the best performance considering ADE and FDE. Compared to vanilla LSTM, the FDE decreased by 20.4%. This indicates a higher accuracy in identifying the moving intention, especially in the going-through cases. And the ADE is lowered by 19.5%, showing a better performance across the whole prediction horizon.

Considering the manoeuvres of all agent types, the going-through cases show the best prediction accuracy while the turning-right cases show the worst. From the table, it can be found that bicycles have the best prediction results compared to cars and trucks.

Table 8-1 Results of model performance

		LSTM_MAP		LSTM_CSP ref		Vanilla LSTM	
		ADE	FDE	ADE	FDE	ADE	FDE
Overall		0.4132	1.0412	0.5026	1.2570	0.5134	1.3083
Car	overall	0.4279	1.1025	0.5214	1.3279	0.5301	1.3724
	gTh	0.4265	1.0778	0.5328	1.3013	0.5163	1.2955
	tl	0.3955	1.0527	0.4655	1.2159	0.5003	1.3204
	tr	0.5049	1.2514	0.6355	1.6239	0.6180	1.5964
Truck	overall	0.4173	1.0201	0.5075	1.2544	0.5287	1.3565
	gTh	0.3698	0.8716	0.4218	0.9392	0.4120	0.9994
	tl	0.3976	0.8806	0.5462	1.2944	0.5721	1.4310
	tr	0.4406	1.1276	0.5067	1.3106	0.5331	1.4035
Bicycle	overall	0.3300	0.7107	0.3968	0.8658	0.4140	0.9335
	gTh	0.2894	0.6177	0.3475	0.7372	0.3662	0.7985
	tl	0.3382	0.7278	0.3972	0.8601	0.4329	1.0019
	tr	0.4739	1.0448	0.5918	1.3889	0.5643	1.3271

8.3.3.2 Performance of Prediction Models in Different Scenarios

The prediction performance of models in different scenarios is shown in Table 8-2. The LSTM_MAP achieves the best performance in all intersections considering ADE and FDE.

Based on the results presented in Table 8-1 and Table 8-2, it can be seen that LSTM_MAP outperforms LSTM_CSP, which integrates RL information and neighbour information respectively. In LSTM_MAP, the proposed agent-map attention effectively identifies the most critical map elements, generating a comprehensive map encoding that reflects their importance. This map encoding not only encodes the moving intentions but also incorporates RL geometry information. This mechanism provides valuable guidance for the TA in its movement, especially in the urban environment where the RLs are well structured.

On the other hand, LSTM_CSP focuses on the influence of the neighbours, which could encode the interaction clues into trajectory prediction. However, in this

project, a rectangular grid is used to organise the neighbours. This structure might not capture the dynamics of interactions, as it primarily focuses on SAs moving in parallel to the TA. It has limitations in handling SAs approaching the TA from the sides, even though these SAs can significantly impact trajectory prediction. Consequently, the results show that LSTM_MAP yields better performance compared to LSTM_CSP.

Table 8-2 Comparative Results on inD measured by ADE/FDE

Method	Intersection A	Intersection B	Intersection C	Overall
LSTM	0.5065/1.3041	0.4895/1.2795	0.5332/1.3270	0.5134/1.3083
LSTM-CSP	0.4815/1.2312	0.4382/1.1440	0.5594/1.3423	0.5026/1.2570
LSTM-MAP	0.4095/1.0332	0.4008/1.0856	0.4237/1.0310	0.4132/1.0412

To have a better visualisation of the displacement errors (DEs) across the whole prediction horizon, the DEs of one actor class at each time stamp are plotted in Figure 8-2.

The prediction model provides robust prediction in Intersection C, where the performances of all agent types and manoeuvres are similar though DEs of turning-right cases are a little bit higher than the two others.

In Intersection A, the prediction results for trucks in different manoeuvres vary a lot. A huge deviation across the prediction horizon could be spotted in the turning-left cases and the general performance in the turning-right cases is good.

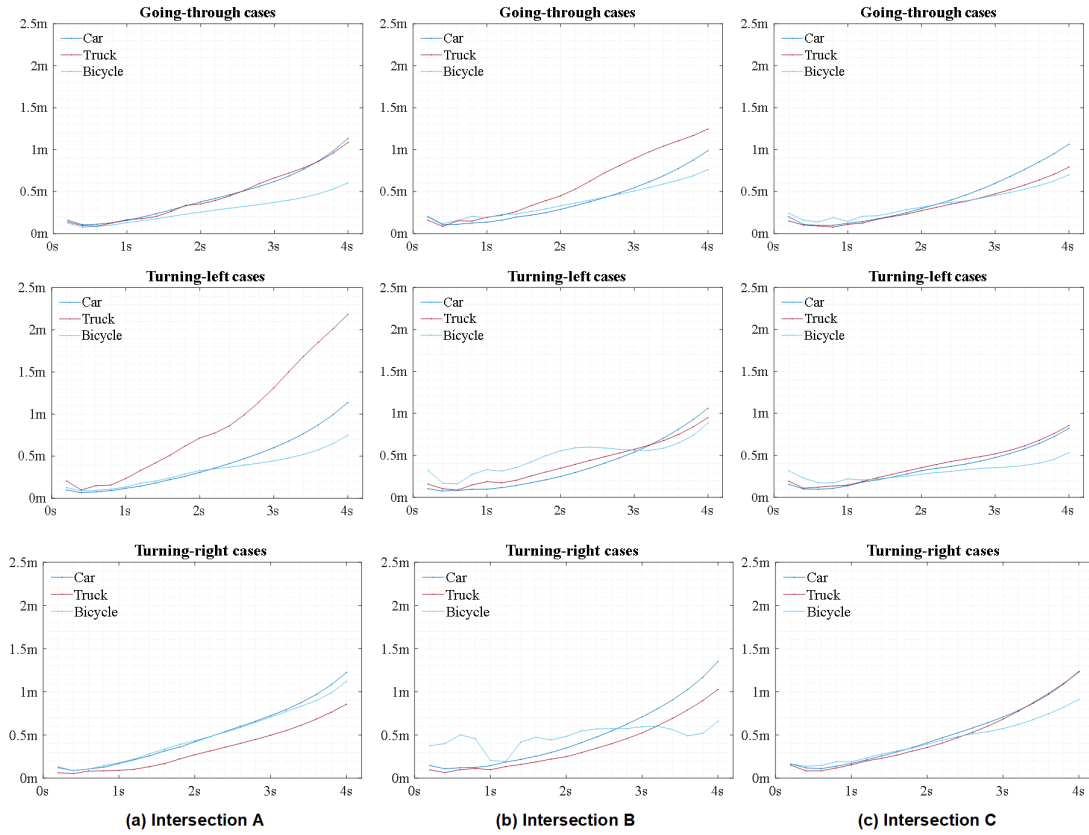


Figure 8-2 Overall performance across the prediction horizon

8.3.3.3 Visualization of Attention Results

The agent-map attention results are visualized in Figure 8-3. It could be seen that AM has helped to select some map elements (Lanelets) that might be most important to the prediction. Generally, the selected map elements tend to have a shape overlapping with the ground truth in the forward driving direction.

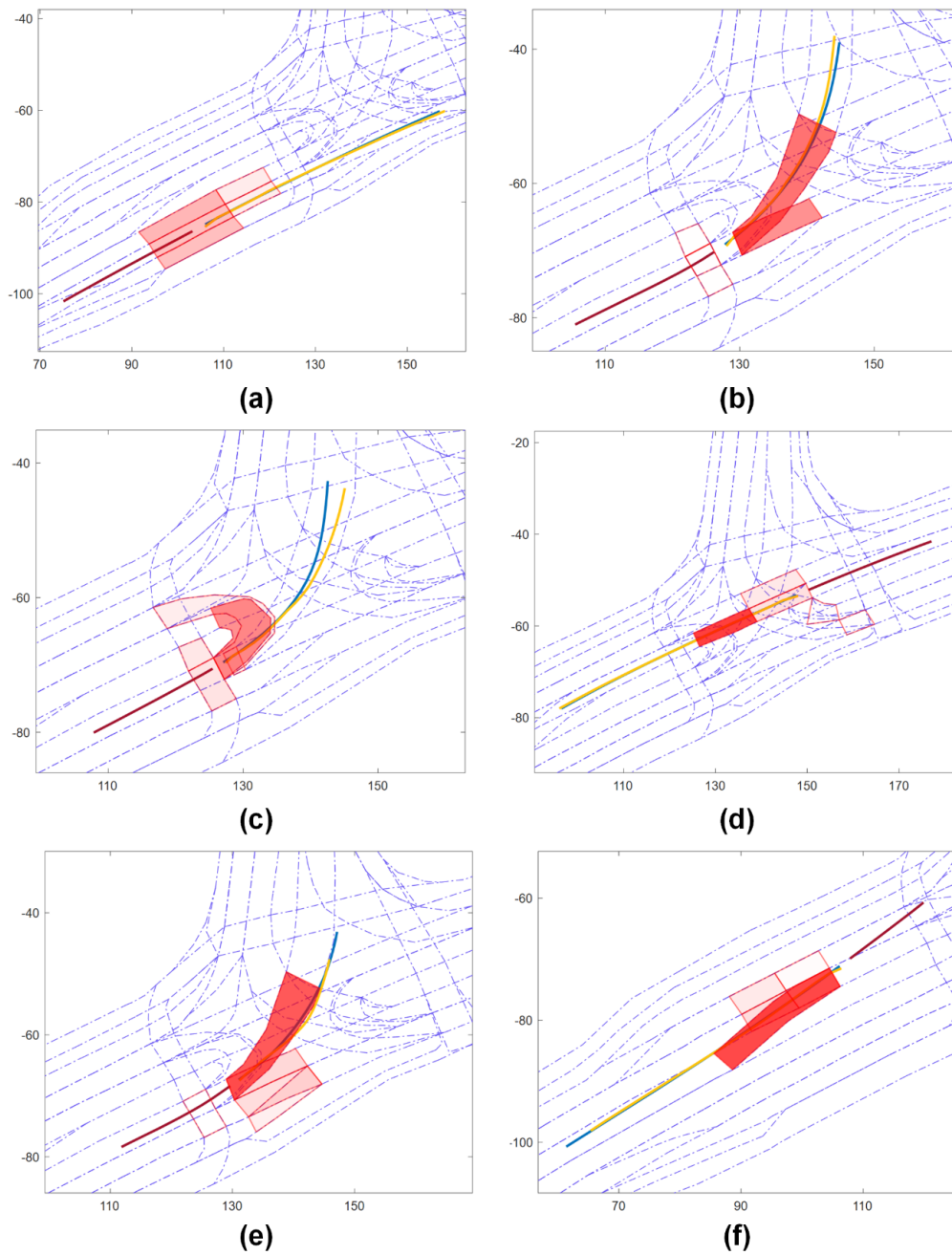


Figure 8-3 Visualization of the attention results. The polygons with darker colours indicate a higher attention score, where more importance is attached compared to the others. The red lines are the historical trajectory of the TA, and the blue and yellow trajectories are the ground truth and predicted trajectories separately.

8.4 Conclusions and Future Work

This chapter presents a trajectory prediction architecture organised around a LSTM-based encoder-decoder structure and inputs of TA's historical status and road layout information. The comparisons of testing results have shown that:

1. Agent-to-map attention has the potential to improve prediction accuracy and provides higher prediction accuracy considering ADE and FED in all scenarios.
2. Agent-to-map attention assigned attention scores to the surrounding map elements and identified the ones that are most significant for the trajectory prediction.

The agent-to-map attention benefits the trajectory prediction by identifying its currently occupied map elements and provides an extension to its driving direction. The map element features, like the centreline, guide the prediction by providing a reference line and the driving direction. This could be partly inferred by the improvement in the FDE, which indirectly shows the moving intention of the agent.

Currently, the map elements are selected by a range search homogeneously regardless of the moving direction and road layout topology. This search method results in more map elements in the traversal direction identified. However, most agents move according to the lane's forward direction instead of its traversal direction. And only by range searching, the exploration of map elements is limited and this might have a negative influence on the prediction.

9 Conclusions and Future Work

9.1 Conclusions

Deep-learning-based trajectory prediction has been explored widely in highway scenarios while little research focuses on urban scenarios due to the limited access to naturalistic driving datasets. The transfer of methods from highway scenarios to urban ones poses great challenges because urban scenarios feature more complex interactions among traffic agents and between agents and road layouts. These complexities call for new approaches to trajectory prediction in urban scenarios.

The motivation of this project is to do research on trajectory prediction in urban scenarios, leveraging naturalistic driving datasets. This study aims to select a trajectory prediction backbone and use the agent interaction and road layout information to perform trajectory prediction.

A comprehensive deep-learning-based trajectory prediction review was conducted. It was generally carried out in terms of trajectory prediction backbone, agent interaction mining and road scene mining. It was revealed that the scope of the prediction components has a strong relationship to the input data formats, which explains why a specific NN architecture is chosen according to the component scope and the input data type and how similar NN architectures could be used in very different components of the prediction model, with different scopes and input structures. Further, appropriate encodings of agent interaction and road scenes improve the prediction accuracy because they provide a better understanding of the traffic scenarios.

For extraction of sufficient samples from the naturalistic driving dataset, tracklets of a fixed temporal horizon are obtained by the sliding window approach. These tracklets are further supplemented with traffic scene information, like the surrounding neighbours of the target agent and road layout information. Considering the road layouts are defined in a sparse representation, a novel centreline extraction method was proposed based on the maximal disks. It has

been proven to provide accurate centrelines and features light data storage and high execution efficiency.

Three of four scenarios in the inD were used to provide track samples and train the trajectory prediction model. An RNN-based encoder-decoder structure was selected as the trajectory prediction backbone. Experimental tests have shown that compared to GRU, LSTM provides a better performance in the trajectory prediction tasks. And a larger size of hidden states in the RNN units tends to provide higher prediction accuracy. Besides, the velocity information of the agents helps to improve the prediction accuracy and the residual connection between the agent's current status to the input of the decoder provides a better inference of the initial status in the prediction horizon.

With the explored trajectory prediction backbone, the agent interaction context was added to the prediction model. A structure replicated from the literature was tested first. It arranges neighbours in a rectangle grid by Euclidean metrics and uses a Convolutional Social Pooling method to extract the agent interaction context. However, the replicated structure did not improve the prediction accuracy in our case. This structure was then tuned to improve the performance. The results showed that it is important to provide the LSTM cell with an appropriate initial hidden state. In our case, the output of the trajectory encoder was used as the initial hidden states of the LSTM cell in the decoder, which is thought to do a good summary of the agent's historical dynamics. This improved structure was further used to include road scene mining.

At last, a trajectory prediction model including road scene mining was proposed. Compared to fine-grained map elements, sparsely-defined road layout elements were used. The map element features were represented by centrepoinets and 1D convolution layers were used to extract map element features. The road scene context was then extracted by agent-map attention. The results showed that road scene mining based on agent-map attention can improve prediction accuracy. The most relevant map elements were assigned with higher attention scores and these map elements had a tendency in the direction of moving.

In summary, the novelties of PhD study are described as:

1. The literature review on DL-based trajectory prediction models is arranged to identify the most successful neural networks (NNs) used in terms of trajectory prediction backbones, agent interaction mining and road scene mining. The analysis is carried out according to the relation between the architecture of the DL-based models for use in vehicle trajectory prediction, the prediction goal, the input information, and the corresponding data formats, which have not been explicitly covered in the literature.
2. A novel approach for the extraction of the road centreline is proposed, based on Maximal Disks (MDs). Instead of resampling the space, MDs are formed directly based on the constraining elements, either segments or their extreme points taken from the boundaries.
3. A small architecture variation has been made on the model replicated from the literature, and this modification provides better performance in agent interaction mining.
4. Sparse road-layout maps are used as map information supplements into the trajectory prediction model for performance enhancement. Agent-map attention is used to extract useful map encodings to enable a map-aware trajectory prediction.

The significance of this project is listed as follows:

1. The review of popular DL-based vehicle trajectory prediction methods reveals the enhanced importance of vehicle historical statuses. Also, the relationships between the scope of the prediction components and the input data formats are explored, as well as the connections with other components in the same prediction framework, including vehicle interaction and road scene mining. This information is crucial to understand complex architectural decisions and to provide guidance for the design of improved solutions.
2. A novel method is proposed to calculate the centreline in a time-efficient and storage-saving way, especially when sparse maps are used. This is important because the sparse maps are a common representation of the road and environment with manual labelling.

3. The comparison of different trajectory prediction model inputs indicates the importance of velocities to prediction accuracy. And the exploration of encoding coupling points guides the concatenation of heterogeneous encodings.
4. The study on map-enhanced trajectory prediction shows the benefits of using sparse road layout maps for prediction performance enhancement.

9.2 Discussion on Generalisation

Generalisation is a key challenge in trajectory prediction. The datasets used for training models are often collected from specific types of scenarios, such as urban intersections, highways, and roundabouts. These scenarios can differ greatly in terms of road layouts, traffic dynamics, and driving behaviours, making it difficult for models trained on one or a few types of scenarios to generalize well across diverse environments. This lack of generalization limits the scalability of trajectory prediction models and restricts their application across different geographic regions and cultural contexts.

Incorporating a wide variety of scenarios with diverse road layouts and traffic conditions into the training process can help reduce overfitting to specific cases. However, it is impractical to cover all possible scenario types exhaustively. Techniques such as transfer learning and one-shot learning are promising to enhance the adaptability of pre-trained models by enabling them to quickly fine-tune and generalize to novel scenarios with minimal additional data.

Since road layouts fundamentally influence the movement patterns of traffic agents, trajectory prediction models can benefit from leveraging map-aware embeddings that encode the relationship between observed traffic agent behaviour and road layout information. Variations in road layout directly lead to distinguishable map-aware embeddings, which can provide shared structural cues across different environments, aiding in generalisation. Furthermore, encoding global road layout information allows the model to differentiate between distinct scenarios and recognise when it is operating in a new domain, thereby improving its ability to adapt to unseen conditions.

Improving generalisation remains crucial for the scalability and deployment of trajectory prediction models across diverse geographic and cultural contexts. By leveraging advanced techniques and leveraging RL information, it is promising to build more robust and adaptable models that are better suited for real-world autonomous driving applications.

9.3 Future Work

In Section 3.6.3, some research gaps have been summarised regarding trajectory prediction. In this section, the following subject areas are also suggested as future works:

1. Construction of more comprehensive naturalistic driving datasets: Currently, trajectory prediction models are trained in a small variety of scenarios, like several roundabouts, cross intersections and highways, limited by the availability of the datasets. These datasets contain distinct manoeuvres, for example, datasets collected in urban might pay more attention to turning instead of lane changing on the highways and even lane-changing manoeuvres are different among different scenarios. Also, some data imbalance exists in these datasets. For example, going-through trajectories might dominate the manoeuvres while few turning trajectories exist. These two issues result in poor performances when the models are exposed to scenarios with different geometries, topologies and traffic characteristics. So, more comprehensive datasets are needed and the prediction model can learn clues from a variety of scenarios and thus produce predictions that are not specific-scenario dependent. Besides, weather conditions are hardly considered. They are also crucial elements for trajectory prediction accuracy. A model trained on sunny days might fail on rainy days because agents' behaviour will be different.
2. Prediction with insufficient scenario information: Assumptions are made that the EV has full observation of the TA with a historical horizon. However, this is not true, especially for the newly perceived TA with a short observation horizon. These TAs might pose a great threat to the EV and it is hard for current trajectory prediction models to estimate their motions

due to a lack of enough historical information. Further, some prediction models deeply use complete road layout information. If the lanes are occupied temporally, for example, due to road obstructions such as accidents or construction, the trained prediction model might struggle to anticipate their motion accurately, as lane-changing manoeuvres in these situations can differ significantly from those in normal driving conditions.

3. Adaptability in varying scenarios: Technologies like few-shot learning and zero-shot learning can be explored to study and improve the generalisation of the prediction model among distinct application scenarios. It is a fact that one prediction model can hardly be generalised among various scenarios due to the complex traffic characteristics. With pre-trained models, solutions can be explored to summarise the characteristics of the traffic scenario and adaptively select the most suitable model to infer the predictions, not limited to meta-learning.
4. Prediction result leverage: The purpose of trajectory prediction is to help the EV better understand the dynamic traffic scenarios during risk assessment and trajectory planning tasks. It is still an open question to make use of the prediction results, especially when the prediction results are represented with Gaussian uncertainties or multi-modal predictions are provided. A probabilistic scenario representation with a long horizon, initialised from the prediction results, introduces massive input information and thus poses challenges to trajectory planning algorithms.
5. Centreline accuracy enhancement: attention could be paid to better resolving the cases of ligature while calculating the lane centreline based on maximal disk methods. This will improve the accuracy of centreline calculations, which will benefit the road layout description and potential path-planning tasks.

REFERENCES

- [1] N. Deo and M. M. Trivedi, "Convolutional Social Pooling for Vehicle Trajectory Prediction," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018.
- [2] D. Ridel, E. Rehder, M. Lauer, C. Stiller and D. Wolf, "A Literature Review on the Prediction of Pedestrian Behavior in Urban Scenarios," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.
- [3] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider and N. Djuric, "Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks," September 2018.
- [4] A. Breuer, S. Elflein, T. Joseph, J.-A. Bolte, S. Homoceanu and T. Fingscheidt, "Analysis of the Effect of Various Input Representations for LSTM-Based Trajectory Prediction," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.
- [5] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley and N. Djuric, "Deep Kinematic Models for Kinematically Feasible Vehicle Trajectory Predictions," August 2019.
- [6] R. Krajewski, J. Bock, L. Kloecker and L. Eckstein, "The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

- [7] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, p. 1231–1237, August 2013.
- [8] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan and J. Hays, *Argoverse: 3D Tracking and Forecasting with Rich Maps*, arXiv, 2019.
- [9] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr and J. Hays, "Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting," January 2023.
- [10] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [11] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens and D. Anguelov, "Large Scale Interactive Motion Forecasting for Autonomous Driving : The Waymo Open Motion Dataset," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [12] A. Zyner, S. Worrall and E. M. Nebot, "ACFR Five Roundabouts Dataset: Naturalistic Driving at Unsignalized Intersections," *IEEE Intelligent Transportation Systems Magazine*, vol. 11, p. 8–18, 2019.
- [13] B. Coifman and L. Li, "A critical evaluation of the Next Generation Simulation (NGSIM) vehicle trajectory dataset," *Transportation Research Part B: Methodological*, vol. 105, p. 362–377, November 2017.

- [14] C. Thiemann, M. Treiber and A. Kesting, "Estimating Acceleration and Lane-Changing Dynamics Based on NGSIM Trajectory Data," *Transportation Research Record: Journal of the Transportation Research Board*, Volume 2088, Pages 90-101 (2008), April 2008.
- [15] R. Huang, G. Zhuo, L. Xiong, S. Lu and W. Tian, "A Review of Deep Learning-Based Vehicle Motion Prediction for Autonomous Driving," *Sustainability*, vol. 15, p. 14716, October 2023.
- [16] A. Robicquet, A. Sadeghian, A. Alahi and S. Savarese, "Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes," in *European Conference on Computer Vision*, 2016.
- [17] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle and M. Tomizuka, "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps," September 2019.
- [18] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," May 2015.
- [19] H. E. RAUCH, T. U. N. G. F. and C. T. STRIEBEL, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, p. 1445–1450, August 1965.
- [20] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater and L. Eckstein, "The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020.
- [21] R. Krajewski, T. Moers, J. Bock, L. Vater and L. Eckstein, "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)*, 2020.

- [22] T. Moers, L. Vater, R. Krajewski, J. Bock, A. Zlocki and L. Eckstein, “The exiD Dataset: A Real-World Trajectory Dataset of Highly Interactive Highway Scenarios in Germany,” in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022.
- [23] A. Breuer, J.-A. Termohlen, S. Homoceanu and T. Fingscheidt, “openDD: A Large-Scale Roundabout Drone Dataset,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020.
- [24] M. Althoff, S. Urban and M. Koschi, “Automatic Conversion of Road Networks from OpenDRIVE to Lanelets,” in *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, 2018.
- [25] H. Ren, H. Gao, H. Chen and G. Liu, “A Survey of Autonomous Driving Scenarios and Scenario Databases,” in *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, 2022.
- [26] M. Haklay and P. Weber, “OpenStreetMap: User-Generated Street Maps,” *IEEE Pervasive Computing*, vol. 7, p. 12–18, October 2008.
- [27] P. Bender, J. Ziegler and C. Stiller, “Lanelets: Efficient map representation for autonomous driving,” in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014.
- [28] A. Barth and U. Franke, “Where will the oncoming vehicle be the next second?,” in *2008 IEEE Intelligent Vehicles Symposium*, 2008.
- [29] S. Ammoun and F. Nashashibi, “Real time trajectory prediction for collision risk estimation between vehicles,” in *2009 IEEE 5th International*

Conference on Intelligent Computer Communication and Processing, 2009.

- [30] P. Karle, M. Geisslinger, J. Betz and M. Lienkamp, "Scenario Understanding and Motion Prediction for Autonomous Vehicles—Review and Comparison," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, p. 16962–16982, October 2022.
- [31] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang and H. Chen, "A Survey on Trajectory-Prediction Methods for Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, p. 652–674, September 2022.
- [32] S. Lefèvre, D. Vasquez and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, July 2014.
- [33] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan and S. Maybank, "A system for learning statistical motion patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, p. 1450–1464, September 2006.
- [34] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings and A. Mouzakitis, "Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, p. 33–47, January 2022.
- [35] V. Bharilya and N. Kumar, *Machine Learning for Autonomous Vehicle's Trajectory Prediction: A comprehensive survey, Challenges, and Future Research Directions*, 2023.
- [36] I. Gomes and D. Wolf, "A Review on Intention-aware and Interaction-aware Trajectory Prediction for Autonomous Vehicles," March 2022.
- [37] E. Schuetz and F. B. Flohr, "A Review of Trajectory Prediction Methods for the Vulnerable Road User," *Robotics*, vol. 13, 2024.

- [38] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65 6, pp. 386-408, 1958.
- [39] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, p. 2222–2232, October 2017.
- [40] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, 2014.
- [41] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012.
- [42] I. Sutskever, O. Vinyals and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," September 2014.
- [43] D. Bahdanau, K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *ArXiv*, vol. 1409, September 2014.
- [44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Attention Is All You Need*, arXiv, 2017.
- [45] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, *Generative Adversarial Networks*, 2014.
- [46] K. Sohn, H. Lee and X. Yan, "Learning Structured Output Representation using Deep Conditional Generative Models," in *Advances in Neural Information Processing Systems*, 2015.

- [47] C. Hegde, S. Dash and P. Agarwal, "Vehicle Trajectory Prediction using GAN," in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2020.
- [48] T. N. Kipf and M. Welling, *Semi-Supervised Classification with Graph Convolutional Networks*, arXiv, 2016.
- [49] R. S. Tomar and S. Verma, "Safety of Lane Change Maneuver Through A Priori Prediction of Trajectory Using Neural Networks," *Netw. Protoc. Algorithms*, vol. 4, pp. 4-21, 2012.
- [50] Q. Liu, B. Lathrop and V. Butakov, "Vehicle lateral position prediction: A small step towards a comprehensive risk assessment system," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014.
- [51] W. Luo, B. Yang and R. Urtasun, "Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net," December 2020.
- [52] S. Casas, W. Luo and R. Urtasun, "IntentNet: Learning to Predict Intention from Raw Sensor Data," January 2021.
- [53] S. Casas, W. Luo and R. Urtasun, "IntentNet: Learning to Predict Intention from Raw Sensor Data," in *Proceedings of The 2nd Conference on Robot Learning*, 2018.
- [54] S. Casas, C. Gulino, R. Liao and R. Urtasun, "Spatially-Aware Graph Neural Networks for Relational Behavior Forecasting from Sensor Data," October 2019.
- [55] B. Yang, W. Luo and R. Urtasun, "PIXOR: Real-Time 3D Object Detection From Point Clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

- [56] F. Althché and A. de La Fortelle, “An LSTM Network for Highway Trajectory Prediction,” January 2018.
- [57] L. Xin, P. Wang, C.-Y. Chan, J. Chen, S. E. Li and B. Cheng, “Intention-aware Long Horizon Trajectory Prediction of Surrounding Vehicles using Dual LSTM Networks,” June 2019.
- [58] A. Zyner, S. Worrall and E. Nebot, “Naturalistic Driver Intention and Path Prediction Using Recurrent Neural Networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, p. 1584–1594, April 2020.
- [59] S. H. Park, B. Kim, C. M. Kang, C. C. Chung and J. W. Choi, “Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture,” February 2018.
- [60] N. Deo and M. M. Trivedi, “Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs,” May 2018.
- [61] F. Giuliari, I. Hasan, M. Cristani and F. Galasso, “Transformer Networks for Trajectory Forecasting,” *CoRR*, vol. abs/2003.08111, 2020.
- [62] Á. Quintanar, D. F. Llorca, I. Parra, R. Izquierdo and M. Á. Sotelo, “Predicting Vehicles Trajectories in Urban Scenarios with Transformer Networks and Augmented Information,” *CoRR*, vol. abs/2106.00559, 2021.
- [63] K. Gao, X. Li, B. Chen, L. Hu, J. Liu, R. Du and Y. Li, “Dual Transformer Based Prediction for Lane Change Intentions and Trajectories in Mixed Traffic Environment,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, p. 6203–6216, June 2023.
- [64] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr and M. Chandraker, “DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents,” April 2017.

- [65] K. Cho, T. Ha, G. Lee and S. Oh, "Deep Predictive Autonomous Driving Using Multi-Agent Joint Trajectory Prediction and Traffic Rules," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [66] A. Dulian and J. C. Murray, "Multi-modal anticipation of stochastic trajectories in a dynamic environment with Conditional Variational Autoencoders," *CoRR*, vol. abs/2103.03912, 2021.
- [67] N. Sriram N., B. Liu, F. Pittaluga and M. Chandraker, "SMART: Simultaneous Multi-Agent Recurrent Trajectory Prediction," *CoRR*, vol. abs/2007.13078, 2020.
- [68] Y. Liu, J. Zhang, L. Fang, Q. Jiang and B. Zhou, "Multimodal Motion Prediction with Stacked Transformers," *CoRR*, vol. abs/2103.11624, 2021.
- [69] Y. Wang, S. Zhao, R. Zhang, X. Cheng and L. Yang, "Multi-Vehicle Collaborative Learning for Trajectory Prediction With Spatio-Temporal Tensor Fusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 236-248, 2022.
- [70] J. Yu, M. Zhou, X. Wang, G. Pu, C. Cheng and B. Chen, "A Dynamic and Static Context-Aware Attention Network for Trajectory Prediction," *ISPRS International Journal of Geo-Information*, vol. 10, p. 336, May 2021.
- [71] Y.-I. Peng, Z.-p. Zhou and L. Li, "Trajectories Prediction of Vehicles at the Intersection Based on LSTM Neural Network," in *CICTP 2019*, 2019.
- [72] S. Dai, L. Li and Z. Li, "Modeling Vehicle Interactions via Modified LSTM Models for Trajectory Prediction," *IEEE Access*, vol. 7, p. 38287–38296, 2019.
- [73] W. Ding, J. Chen and S. Shen, "Predicting Vehicle Behaviors Over An Extended Horizon Using Behavior Interaction Network," March 2019.

- [74] K. Zhang, L. Zhao, C. Dong, L. Wu and L. Zheng, "AI-TP: Attention-Based Interaction-Aware Trajectory Prediction for Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 8, p. 73–83, January 2023.
- [75] Y. Gao, J. Fu, W. Feng, T. Xu and K. Yang, "Surrounding vehicle trajectory prediction under mixed traffic flow based on graph attention network," *Physica A: Statistical Mechanics and its Applications*, vol. 639, p. 129643, 2024.
- [76] Z. Sheng, Z. Huang and S. Chen, "Kinematics-Aware Multigraph Attention Network with Residual Learning for Heterogeneous Trajectory Prediction," *Journal of Intelligent and Connected Vehicles*, vol. 7, pp. 138-150, 2024.
- [77] X. Li, X. Ying and M. C. Chuah, "GRIP: Graph-based Interaction-aware Trajectory Prediction," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.
- [78] Y. Ren, Z. Lan, L. Liu and H. Yu, "EMSIN: Enhanced Multi-Stream Interaction Network for Vehicle Trajectory Prediction," *IEEE Transactions on Fuzzy Systems*, pp. 1-15, 2024.
- [79] Z. Li, C. Lu, Y. Yi and J. Gong, "A Hierarchical Framework for Interactive Behaviour Prediction of Heterogeneous Traffic Participants Based on Graph Neural Network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 9102-9114, 2022.
- [80] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang and D. Manocha, "TrafficPredict: Trajectory Prediction for Heterogeneous Traffic-Agents," November 2018.
- [81] Y. Tang, H. He and Y. Wang, "Hierarchical vector transformer vehicle trajectories prediction with diffusion convolutional neural networks," *Neurocomputing*, vol. 580, p. 127526, 2024.

- [82] H. Sadid and C. Antoniou, "Dynamic Spatio-temporal Graph Neural Network for Surrounding-aware Trajectory Prediction of Autonomous Vehicles," *IEEE Transactions on Intelligent Vehicles*, pp. 1-14, 2024.
- [83] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu and F. Moutarde, "HOME: Heatmap Output for future Motion Estimation," May 2021.
- [84] L. D. W. Z. X. Y. Zhiyu Yang and Y. Han, "Vehicle trajectory prediction based on attention optimized with real-scene sampling," *Systems Science & Control Engineering*, vol. 12, p. 2347889, 2024.
- [85] H. Song, D. Luan, W. Ding, M. Y. Wang and Q. Chen, "Learning to Predict Vehicle Trajectories with Model-based Planning," March 2021.
- [86] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu and F. Moutarde, *GOHOME: Graph-Oriented Heatmap Output for future Motion Estimation*, arXiv, 2021.
- [87] K. Messaoud, I. Yahiaoui, A. Verroust-Blondet and F. Nashashibi, "Attention Based Vehicle Trajectory Prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 6, p. 175–185, March 2021.
- [88] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [89] J. W. Junyu Cai and A. Li, "Multi-head attention-based intelligent vehicle lane change decision and trajectory prediction model in highways," *Journal of Intelligent Transportation Systems*, vol. 0, p. 1–18, 2024.
- [90] C. Li, Z. Liu, S. Lin, Y. Wang and X. Zhao, "Intention-convolution and hybrid-attention network for vehicle trajectory prediction," *Expert Systems with Applications*, vol. 236, p. 121412, 2024.

- [91] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang and Y. N. Wu, "Multi-Agent Tensor Fusion for Contextual Trajectory Prediction," April 2019.
- [92] M. Hasan, A. Solernou, E. Paschalidis, H. Wang, G. Markkula and R. Romano, *Maneuver-Aware Pooling for Vehicle Trajectory Prediction*, arXiv, 2021.
- [93] J. Li, H. Shi, Y. Guo, G. Han, R. Yu and X. Wang, "TraGCAN: Trajectory Prediction of Heterogeneous Traffic Agents in IoV Systems," *IEEE Internet of Things Journal*, vol. 10, p. 7100–7113, April 2023.
- [94] C. Fei, X. He and X. Ji, "Multi-modal vehicle trajectory prediction based on mutual information," *IET Intelligent Transport Systems*, vol. 14, p. 148–153, February 2020.
- [95] H. Min, X. Xiong, P. Wang and Z. Zhang, "A Hierarchical LSTM-Based Vehicle Trajectory Prediction Method Considering Interaction Information," *Automotive Innovation*, vol. 7, p. 71–81, 2024.
- [96] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin and J. Schneider, "Short-term Motion Prediction of Traffic Actors for Autonomous Driving using Deep Convolutional Networks," August 2018.
- [97] N. Deo and M. M. Trivedi, "Trajectory Forecasts in Unknown Environments Conditioned on Grid-Based Plans," *ArXiv*, vol. abs/2001.00735, 2020.
- [98] K. Messaoud, N. Deo, M. M. Trivedi and F. Nashashibi, "Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021.
- [99] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider and N. Djuric, "Multimodal Trajectory Predictions for

Autonomous Driving using Deep Convolutional Networks,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

- [100] L. Hou, S. E. Li, B. Yang, Z. Wang and K. Nakano, “Integrated Graphical Representation of Highway Scenarios to Improve Trajectory Prediction of Surrounding Vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 8, p. 1638–1651, February 2023.
- [101] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li and C. Schmid, “VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation,” May 2020.
- [102] B. Kim, S. H. Park, S. Lee, E. Khoshimjonov, D. Kum, J. Kim, J. S. Kim and J. W. Choi, *LaPred: Lane-Aware Prediction of Multi-Modal Future Trajectories of Dynamic Agents*, arXiv, 2021.
- [103] N. Deo, E. Wolff and O. Beijbom, “Multimodal Trajectory Prediction Conditioned on Lane-Graph Traversals,” in *Proceedings of the 5th Conference on Robot Learning*, 2022.
- [104] L. Zhang, P. Li, J. Chen and S. Shen, “Trajectory Prediction with Graph-based Dual-scale Context Fusion,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.
- [105] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng and R. Urtasun, “Learning Lane Graph Representations for Motion Forecasting,” in *Computer Vision – ECCV 2020*, Springer International Publishing, 2020, p. 541–556.
- [106] W. Zeng, M. Liang, R. Liao and R. Urtasun, “LaneRCNN: Distributed Representations for Graph-Centric Motion Forecasting,” January 2021.
- [107] P. Bhattacharyya, C. Huang and K. Czarnecki, “SSL-Lanes: Self-Supervised Learning for Motion Forecasting in Autonomous Driving,” June 2022.

- [108] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom and E. M. Wolff, "CoverNet: Multimodal Behavior Prediction using Trajectory Sets," November 2019.
- [109] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal and others, "Scene transformer: A unified architecture for predicting multiple agent trajectories," *arXiv preprint arXiv:2106.08417*, 2021.
- [110] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu and F. Moutarde, *THOMAS: Trajectory Heatmap Output with learned Multi-Agent Sampling*, arXiv, 2021.
- [111] H. Guo, Q. Meng, X. Zhao, J. Liu, D. Cao and H. Chen, "Map-enhanced generative adversarial trajectory prediction method for automated vehicles," *Information Sciences*, vol. 622, p. 1033–1049, April 2023.
- [112] C. Luo, L. Sun, D. Dabiri and A. Yuille, "Probabilistic Multi-modal Trajectory Prediction with Lane Attention for Autonomous Vehicles," July 2020.
- [113] X. Mo, Y. Xing, H. Liu and C. Lv, "Map-adaptive multimodal trajectory prediction using hierarchical graph neural networks," *IEEE Robotics and Automation Letters*, 2023.
- [114] M. Yang, B. Zhang, T. Wang, J. Cai, X. Weng, H. Feng and K. Fang, "Vehicle Interactive Dynamic Graph Neural Network Based Trajectory Prediction for Internet of Vehicles," *IEEE Internet of Things Journal*, pp. 1-1, 2024.
- [115] H. Liao, Z. Li, H. Shen, W. Zeng, D. Liao, G. Li and C. Xu, "Bat: Behavior-aware human-like trajectory prediction for autonomous driving," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

- [116] H. Liao, Y. Li, Z. Li, C. Wang, Z. Cui, S. E. Li and C. Xu, "A Cognitive-Based Trajectory Prediction Approach for Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, 2024.
- [117] M. Liu, H. Cheng, L. Chen, H. Broszio, J. Li, R. Zhao, M. Sester and M. Y. Yang, "LAformer: Trajectory Prediction for Autonomous Driving with Lane-Aware Scene Constraints," February 2023.
- [118] X. Wang, T. Su, F. Da and X. Yang, "ProphNet: Efficient Agent-Centric Motion Forecasting with Anchor-Informed Proposals," March 2023.
- [119] Z. Zhou, L. Ye, J. Wang, K. Wu and K. Lu, "HiVT: Hierarchical Vector Transformer for Multi-Agent Motion Prediction," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [120] B. He and Y. Li, "Multi-future Transformer: Learning diverse interaction modes for behaviour prediction in autonomous driving," *IET Intelligent Transport Systems*, vol. 16, p. 1249–1267, June 2022.
- [121] Z. Lan, Y. Jiang, Y. Mu, C. Chen and S. E. Li, "SEPT: Towards Efficient Scene Representation Learning for Motion Prediction," September 2023.
- [122] X. Gao, X. Jia, Y. Li and H. Xiong, "Dynamic Scenario Representation Learning for Motion Forecasting with Heterogeneous Graph Convolutional Recurrent Networks," March 2023.
- [123] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov and S. Zagoruyko, "End-to-End Object Detection with Transformers," *CoRR*, vol. abs/2005.12872, 2020.
- [124] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat and B. Sapp, "Wayformer: Motion Forecasting via Simple & Efficient Attention Networks," July 2022.

- [125] Z. Zhou, J. Wang, Y.-H. Li and Y.-K. Huang, "Query-Centric Trajectory Prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [126] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov and B. Sapp, "MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction," November 2021.
- [127] R. UNIVERSITY, "THE INTERSECTION DRONE DATASET," [Online]. Available: <https://levelxdata.com/ind-dataset/>. [Accessed 30 01 2024].
- [128] M. Montanino and V. Punzo, "Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns," *Transportation Research Part B: Methodological*, vol. 80, p. 82–106, October 2015.
- [129] M. Geng, Z. Cai, Y. Zhu, X. Chen and D.-H. Lee, "Multimodal Vehicular Trajectory Prediction With Inverse Reinforcement Learning and Risk Aversion at Urban Unsignalized Intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, p. 12227–12240, November 2023.
- [130] M. Geng, Y. Chen, Y. Xia and X. (. Chen, "Dynamic-learning spatial-temporal Transformer network for vehicular trajectory prediction at urban intersections," *Transportation Research Part C: Emerging Technologies*, vol. 156, p. 104330, November 2023.
- [131] C.-F. Lin, A. G. Ulsoy and D. J. LeBlanc, "Vehicle dynamics and external disturbance estimation for vehicle path prediction," *IEEE Transactions on Control Systems Technology*, vol. 8, p. 508–518, May 2000.
- [132] R. Rajamani, *Vehicle Dynamics and Control*, Springer US, 2012.

- [133] R. Schubert, E. Richter and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *2008 11th International Conference on Information Fusion*, 2008.
- [134] N. Kaempchen, K. Weiss, M. Schaefer and K. C. J. Dietmayer, "IMM object tracking for high dynamic driving maneuvers," in *IEEE Intelligent Vehicles Symposium, 2004*, 2004.
- [135] B. Jin, B. Jiu, T. Su, H. Liu and G. Liu, "Switched Kalman filter-interacting multiple model algorithm based on optimal autoregressive model for manoeuvring target tracking," *IET Radar, Sonar & Navigation*, vol. 9, p. 199–209, February 2015.
- [136] H. Dyckmanns, R. Matthaei, M. Maurer, B. Lichte, J. Effertz and D. Stiker, "Object tracking in urban intersections based on active use of a priori knowledge: Active interacting multi model filter," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011.
- [137] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, p. 1735–1780, November 1997.
- [138] S. Yang, X. Yu and Y. Zhou, "LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example," in *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, 2020.
- [139] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.
- [140] A. Geiger, M. Lauer, F. Moosmann, B. Ranft, H. Rapp, C. Stiller and J. Ziegler, "Team AnnieWAY\textquotesingles Entry to the 2011 Grand

Cooperative Driving Challenge,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, p. 1008–1017, September 2012.

- [141] E. Murciego, C. G. Huélamo, R. Barea, L. M. Bergasa, E. Romera, J. F. Arango, M. Tradacete and Á. Sáez, “Topological Road Mapping for Autonomous Driving Applications,” in *Advances in Intelligent Systems and Computing*, Springer International Publishing, 2018, p. 257–270.
- [142] A. e.V., “ASAM OpenDRIVE,” [Online]. Available: <https://www.asam.net/standards/detail/opendrive/>. [Accessed 25 04 2023].
- [143] W. Shi, Z. Miao and J. Debayle, “An Integrated Method for Urban Main-Road Centerline Extraction From Optical Remotely Sensed Imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, p. 3359–3372, June 2014.
- [144] R. Liu, J. Song, Q. Miao, P. Xu and Q. Xue, “Road centerlines extraction from high resolution images based on an improved directional segmentation and road probability,” *Neurocomputing*, vol. 212, p. 88–95, November 2016.
- [145] Y. Shen, T. Ai and M. Yang, “Extracting Centerlines From Dual-Line Roads Using Superpixel Segmentation,” *IEEE Access*, vol. 7, p. 15967–15979, 2019.
- [146] C. Sujatha and D. Selvathi, “Connected component-based technique for automatic extraction of road centerline in high resolution satellite images,” *EURASIP Journal on Image and Video Processing*, vol. 2015, March 2015.
- [147] F. Xiao, L. Tong, Y. Li, S. Luo and J. A. Benediktsson, “A General Spline-Based Method for Centerline Extraction from Different Segmented Road Maps in Remote Sensing Imagery,” *Remote Sensing*, vol. 14, p. 2074, April 2022.

- [148] G. Cheng, F. Zhu, S. Xiang and C. Pan, "Road Centerline Extraction via Semisupervised Segmentation and Multidirection Nonmaximum Suppression," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, p. 545–549, April 2016.
- [149] T. Alshaikhli, W. Liu and Y. Maruyama, "Simultaneous Extraction of Road and Centerline from Aerial Images Using a Deep Convolutional Neural Network," *ISPRS International Journal of Geo-Information*, vol. 10, p. 147, March 2021.
- [150] W. Yujun, H. Xiangyun and G. Jinqi, "End-to-End Road Centerline Extraction via Learning a Confidence Map," in *2018 10th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)*, 2018.
- [151] M. Klischat, E. I. Liu, F. Holtke and M. Althoff, "Scenario Factory: Creating Safety-Critical Traffic Scenarios for Automated Vehicles," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020.
- [152] B. Liao, S. Chen, B. Jiang, T. Cheng, Q. Zhang, W. Liu, C. Huang and X. Wang, *Lane Graph as Path: Continuity-preserving Path-wise Modeling for Online Lane Graph Construction*, 2023.
- [153] X. Xiong, Y. Liu, T. Yuan, Y. Wang, Y. Wang and H. Zhao, *Neural Map Prior for Autonomous Driving*, 2023.
- [154] S. Chen, Y. Zhang, B. Liao, J. Xie, T. Cheng, W. Sui, Q. Zhang, C. Huang, W. Liu and X. Wang, *VMA: Divide-and-Conquer Vectorized Map Annotation System for Large-Scale Driving Scene*, 2023.
- [155] Y. Liu, T. Yuan, Y. Wang, Y. Wang and H. Zhao, *VectorMapNet: End-to-end Vectorized HD Map Learning*, 2023.

- [156] B. Liao, S. Chen, Y. Zhang, B. Jiang, Q. Zhang, W. Liu, C. Huang and X. Wang, *MapTRv2: An End-to-End Framework for Online Vectorized HD Map Construction*, 2023.
- [157] M. G. Wagner, "Real-time thinning algorithms for 2D and 3D images using GPU processors," *Journal of Real-Time Image Processing*, vol. 17, p. 1255–1266, May 2019.
- [158] M. McAllister and J. Snoeyink, "Medial Axis Generalization of River Networks," *Cartography and Geographic Information Science*, vol. 27, p. 129–138, January 2000.
- [159] H. Z. Selvi, I. O. Bildirici and M. Yerci, "Triangulation Method for Area-line Geometry-type Changes in Map Generalisation," *The Cartographic Journal*, vol. 47, p. 157–163, May 2010.
- [160] A. Golly and J. M. Turowski, "Deriving principal channel metrics from bank and long-profile geometry with the R package cmgo," *Earth Surface Dynamics*, vol. 5, p. 557–570, September 2017.
- [161] S. A. K. Tejenaki, H. Ebadi and A. Mohammadzadeh, "A new hierarchical method for automatic road centerline extraction in urban areas using LIDAR data," *Advances in Space Research*, vol. 64, p. 1792–1806, November 2019.
- [162] L. Lam, S.-W. Lee and C. Y. Suen, "Thinning methodologies-a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, p. 869–885, 1992.
- [163] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, p. 236–239, March 1984.
- [164] L. B. Boudaoud, A. Sider and A. Tari, "A new thinning algorithm for binary images," in *2015 3rd International Conference on Control, Engineering & Information Technology (CEIT)*, 2015.

- [165] Z. Chen, X. Yin, L. Lin, G. Shi and J. Mo, "Centerline extraction by neighborhood-statistics thinning for quantitative analysis of corneal nerve fibers," *Physics in Medicine & Biology*, vol. 67, p. 145005, July 2022.
- [166] L. B. Boudaoud, B. Solaiman and A. Tari, "A modified ZS thinning algorithm by a hybrid approach," *The Visual Computer*, vol. 34, p. 689–706, May 2017.
- [167] T. Strutz, *The Distance Transform and its Computation*, arXiv, 2021.
- [168] Y. Ge and J. M. Fitzpatrick, "On the generation of skeletons from discrete Euclidean distance maps," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, p. 1055–1066, 1996.
- [169] S. Koka, K. Anada, Y. Nakayama, K. Sugita, T. Yaku and R. Yokoyama, "A Comparison of Ridge Detection Methods for DEM Data," in *2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2012.
- [170] C. W. Niblack, P. B. Gibbons and D. W. Capson, "Generating skeletons and centerlines from the distance transform," *CVGIP: Graphical Models and Image Processing*, vol. 54, p. 420–437, September 1992.
- [171] H. Blum and R. N. Nagel, "Shape description using weighted symmetric axis features," *Pattern Recognition*, vol. 10, p. 167–180, January 1978.
- [172] W.-P. Choi, K.-M. Lam and W.-C. Siu, "Extraction of the Euclidean skeleton based on a connectivity criterion," *Pattern Recognition*, vol. 36, p. 721–729, March 2003.
- [173] X. Bai, L. Latecki and W.-y. Liu, "Skeleton Pruning by Contour Partitioning with Discrete Curve Evolution," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, p. 449–462, March 2007.

- [174] D. T. Lee and B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation," *International Journal of Computer & Information Sciences*, vol. 9, p. 219–242, June 1980.
- [175] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, p. 269–271, December 1959.
- [176] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," June 2014.
- [177] A. Graves, "Long Short-Term Memory," in *Studies in Computational Intelligence*, Springer Berlin Heidelberg, 2012, p. 37–45.
- [178] J. Pan, H. Sun, K. Xu, Y. Jiang, X. Xiao, J. Hu and J. Miao, "Lane-Attention: Predicting Vehicles' Moving Trajectories by Learning Their Attention Over Lanes," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

