Vehicle Trajectory Prediction for Safe Navigation of Autonomous Vehicles

Heriot-Watt University

SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Saptarishi Mukherjee

Supervised by: Prof. Andrew Wallace and Dr. Sen Wang

February 2022

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Abstract

Trajectory prediction of the other road users in the vicinity of an autonomous vehicle is important for safe navigation in dense traffic. Once an autonomous vehicle anticipates how the other road actors will react in the near future, path planning is a lot more simpler and safer. Moreover, the knowledge of future movement of other road actors allows control of sudden jerks in the planned ego vehicle's path and thus makes travel smoother. This trajectory prediction stage can be used at any level, from restricted driver assistance to full vehicle autonomy.

In this thesis two novel trajectory prediction models have been developed. In the first model, the spatio-temporal features that form the basis of behaviour prediction were captured using a Convolutional Long Short Term Memory (Conv-LSTM) neural network architecture consisting of three modules: 1) Interaction Learning to capture the motion of and interaction with surrounding cars, 2) Temporal Learning to identify the dependency on past movements and 3) Motion Learning to convert the extracted features from these two modules into future positions. In addition, a novel feedback scheme was introduced in which the current predicted positions of each car are leveraged to update future motion, encapsulating the effect of the surrounding cars. In the second model a conventional Long Short Term Memory (LSTM) cell based encoder-decoder architecture was developed which uses not only the historical observations but also the associated map features. Moreover, unlike existing architectures, the proposed method incorporates and updates the surrounding vehicle information in both the encoder and decoder, making use of dynamically predicted new data for accurate prediction in longer time horizons. This seamlessly performs four tasks: first, it encodes a feature given the past observations, second, it estimates future maneuvers given the encoded state, third, it predicts the future motion given the estimated maneuvers and the initially encoded states, and fourth, it estimates future trajectory given the encoded state and the predicted maneuvers and motions.

Both the developed models were evaluated extensively on two publicly available datasets which include both multi-lane highway and signalled intersections, to benchmark the prediction accuracy with the state-of-the-art models. Later, the conventional encoder-decoder model was also evaluated with a newly collected "Ra-

diate" dataset which includes two intersections, the Kingussie T-junction and the Edinburgh four-way junction, both without traffic signals. The accuracy of the predicted trajectories on the benchmark datasets are comparable with state-of-the-art methods. Moreover, evaluation on the latter dataset ("Radiate") made it possible to understand better the effect of inter-vehicle interactions on future motion without any influence from mandatory traffic signals.

Acknowledgements

This has been an wonderful journey. It most certainly would not have been possible without the endless effort from my supervisor Prof. Andrew M. Wallace (Andy) and my co-supervisor Dr. Sen Wang. A big thank you to Jaguar Land Rover, Heriot-Watt University and EPSRC for their financial support through out the entire tenure.

Thank you to Ali, Marcel, Zhiyan and the rest of the team working on the Jaguar Land Rover and related projects. Together we handled many issues starting from critical sensor calibration to data collection at mid-night to collect data in adverse light conditions.

Thank you to all my friends, Ankit, Sobhan and Abhijeet for always cheering me up whenever I felt a bit down. Last but not the least, a big thank you to my parents and my elder sister (Khushi) for their continuous support and encouragement through out this journey.

Research Thesis Submission

Name*:*	Saptarshi Mukherjee		
School:	EPS		
Version: *(i.e. First, Resubmission, Final)*	Final	Degree Sought:	Ph.D.

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1. The thesis embodies the results of my own work and has been composed by myself
2. Where appropriate, I have made acknowledgement of the work of others
3. The thesis is the correct version for submission and is the same version as any electronic versions submitted*.
4. My thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5. I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.
6. I confirm that the thesis has been verified against plagiarism via an approved plagiarism detection application e.g. Turnitin.

ONLY for submissions including published works

7. Where the thesis contains published outputs under Regulation 6 (9.1.2) or Regulation 43 (9) these are accompanied by a critical review which accurately describes my contribution to the research and, for multi-author outputs, a signed declaration indicating the contribution of each author (complete)
8. Inclusion of published outputs under Regulation 6 (9.1.2) or Regulation 43 (9) shall not constitute plagiarism.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate*:*	Saptarshi Mukherjee	Date:	25/02/2022

Submission

Submitted By *(name in capitals):*	SAPTARSHI MUKHERJEE
Signature of Individual Submitting:	Saptarshi Mukherjee
Date Submitted:	25/02/2022

For Completion in the Student Service Centre (SSC)

Limited Access	Requested	Yes		No		Approved	Yes		No	
*E-thesis Submitted (**mandatory for final theses)***										
Received in the SSC by *(name in capitals):*						Date:				

Inclusion of Published Works

Declaration

This thesis contains one or more multi-author published works. In accordance with Regulation 6 (9.1.2) I hereby declare that the contributions of each author to these publications is as follows:

Citation details	S. Mukherjee, A. M. Wallace and S. Wang, ``Predicting Vehicle Behavior Using Automotive Radar and Recurrent Neural Networks'' in IEEE Open Journal of Intelligent Transportation Systems, vol. 2, pp. 254-268, 2021, doi: 10.1109/OJITS.2021.3105920
Author 1	Technical contribution and main author
Author 2	Technical, paper feedback and corrections
Author 3	Technical, paper feedback and corrections
Signature:	Saptarshi Mukherjee
Date:	25/02/2022

Citation details	S. Mukherjee, S. Wang and A. M. Wallace, ``Interacting Vehicle Trajectory Prediction with Convolutional Recurrent Neural Networks'' 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 4336-4342, doi: 10.1109/ICRA40945.2020.9196807
Author 1	Technical contribution and main author
Author 2	Technical, paper feedback and corrections
Author 3	Technical, paper feedback and corrections
Signature:	Saptarshi Mukherjee
Date:	25/02/2022

Citation details	M. Sheeny, E. D. Pellegrin, S. Mukherjee, A. Ahrabian, S. Wang, & A. M. Wallace, (2021, May). RADIATE: A radar dataset for automotive perception in bad weather. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1-7). IEEE.
Author 1	Technical contribution and main author
Author 2	Data annotation
Author 3	Test vehicle setup and data collection software architecture
Author 4	Data collection
Author 5	Technical, paper feedback and corrections
Author 6	Technical, paper feedback and corrections
Signature:	Saptarshi Mukherjee
Date:	25/02/2022

Citation details	A. M. Wallace, S. Mukherjee, B. Toh, A. Ahrabian, ``Combining automotive radar and LiDAR for surface detection in adverse conditions'', IET Radar, Sonar and Navigation, vol. 15, no. 4, pp. 359-369, 2021, doi:10.1049/rsn2.12042
Author 1	Technical contribution and main author
Author 2	Data preparation and experiments
Author 3	Technical contribution in the experiments
Author 4	Technical contribution in the experiments
Signature:	Saptarshi Mukherjee
Date:	25/02/2022

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ADAS	Advanced Driver-Assistance Systems
APM	Advanced Path Measurement
ANN	Artificial Neural Network
AND	Advanced Navigation Driver
BEV	Bird's Eye View
CCTV	Closed-Circuit Television
CNN	Convolutional Neural Network
CS-LSTM	Convolutional Social Pooling LSTM
CV	Constant Velocity
CTRA	Constant Turn Rate and Acceleration
CTRV	Constant Turn Rate and Velocity
DBN	Dynamic Bayesian Networks
DC	Direct Current
DHW	Distance Headway
DNN	Deep Neural Networks
DoA	Direction of Arrival
DTW	Dynamic Time Warping
ELU	Exponential Linear Unit
ES	Encoded States
FC	Fully Connected
FMCW	Frequency-Modulated Continuous Wave
FFT	Fast Fourier Transform
GAN	Generative Adversarial Networks
G-HMM	Growing-Hidden Markov Model
GIS	Geographic Information System
GMM	Gaussian Mixture Model
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GRU	Gated Recurrent Unit
HGV	Heavy Goods Vehicles
HMM	Hidden Markov Model
IDM	Intelligent Driver Model
IMM	Interacting Multiple Model
IMU	Inertial Measurement Unit
ISAR	Inverse Synthetic-Aperture Radar
LC	Lane Change
LCSS	Longest Common Subsequence

LGV	Large Goods Vehicles
LiDAR	Light Detection and Ranging
LK	Lane Keeping
LLC	Left Lane Change
LMV	Light Motor Vehicles
LSTM	Long Short Term Memory
MATF	Multi-Agent Tensor Fusion
MDN	Mixture Density Networks
MIDM	Modified Intelligent Driver Model
MIMO	Multiple Input Multiple Output
MSFM	Modified Social Force Model
NGSIM	Next Generation Simulation
OGM	Occupancy Grid Map
OS	Operating System
POMDP	Partially Observable Markov Decision Process
RADIATE	RAdar Dataset In Adverse weaThEr (RADIATE)
RCA	Radio Corporation of America
ReLU	Rectified Linear Unit
RF	Radio Frequency
RGB	Red Green Blue
RLC	Right Lane Change
RMSE	Root Mean Square Error
RRNN	Relational Recurrent Neural Network
RNN	Recurrent Neural Network
ROI	Region of Interest
ROS	Robot Operating System
ROS	Real-Time Kinematic
SAE	Society of Automotive Engineers
SAR	Synthetic-Aperture Radar
SeqToSeq	Sequence-Input-Single-Output
SeqToSeq	sequence-input-sequence-output
SFM	Social Force Model
SLAM	Simultaneous Localization and Mapping
SV	Surrounding Vehicle
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor Embedding
THW	Time Headway (THW)
THW	Time to Collision (TTC)
TTLC	Time to Lane Change
TV	Target Vehicle
UAV	Unmanned Aerial Vehicle
V2V	Vehicle to Vehicle
V-LSTM	Vanilla LSTM
WAE	Wasserstein Auto-Encoder
WHO	World Health Organization

Nomenclature

Features	
β	Lane number
μ	Lane movement rule
$\mathbf{pos} = (x, y)$	Vehicle position
$\mathbf{v} = (\dot{x}, \dot{y})$	Vehicle motion
d	Distance from the lane
j	Distance from the junction
Input/Output	
$\mathbf{D}_t$	Decoder Input at t
$\mathbf{m}$	$[\mathrm{P}^s, P^l, P^r]$
$\mathbf{O}_t$	Encoder Input at t
C	Number of Columns in each OGM
F	Prediction sequence length
$GridLen$	Physical length of each OGM in real world (feet)
$GridWid$	Physical width of each OGM in real world (feet)
h	Input sequence length
OGM_t^n	Occupancy Grid Map at time t for vehicle n
P^l	Probability of left maneuver
P^r	Probability of right maneuver
P^s	Probability of straight maneuver
R	Number of Rows in each OGM
t	Current time
Vehicles	
Len	Physical Length of the vehicle in feet
N	Number of vehicles in the scene
ns	Considered SVs during prediction
Wid	Physical Width of the vehicle in feet

List of Publications

- **S. Mukherjee**, A. M. Wallace and S. Wang, "Predicting Vehicle Behavior Using Automotive Radar and Recurrent Neural Networks" in IEEE Open Journal of Intelligent Transportation Systems, vol. 2, pp. 254-268, 2021, doi: 10.1109/OJITS.2021.3105920.

- **S. Mukherjee**, S. Wang and A. M. Wallace, "Interacting Vehicle Trajectory Prediction with Convolutional Recurrent Neural Networks" 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 4336-4342, doi: 10.1109/ICRA40945.2020.9196807.

- M. Sheeny, E. D. Pellegrin, **S. Mukherjee**, A. Ahrabian, S. Wang, A. M. Wallace, (2021, May). RADIATE: A radar dataset for automotive perception in bad weather. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1-7). IEEE.

- A. M. Wallace, **S. Mukherjee**, B. Toh, A. Ahrabian, "Combining automotive radar and LiDAR for surface detection in adverse conditions", IET Radar, Sonar and Navigation, vol. 15, no. 4, pp. 359-369, 2021, doi:10.1049/rsn2.12042

Chapter 1

Introduction

According to the World Health Organization (WHO) [171], approximately 1.35 million people die each year due to road accidents, approximately 3,287 deaths per day. In addition, another 20 to 50 million people suffer non-fatal injuries per year, many suffering a permanent disability. To reduce these injuries and fatalities various improvements have been made to consumer vehicles including multiple airbags, anti-lock brakes, crumple zones, seat-belt pre-tensioners, traction control, rigid passenger cabins, and advanced driver assistance systems including lane departure warnings, fatigue detection, forward collision warning systems and blind spot monitors, to name a few. Despite these advanced driving assisting features the fatality rate is still high and in over 90% of cases it is due to human error [129], [154], [155]. Aware of the fatality rates and the contribution of human error, researchers have tried to develop self-driving or autonomous cars in which the human driver is not necessary, including computational sensing and an automated control system.

The processing loop employed in autonomous vehicle navigation includes six different progressive stages, sensing, perception, localization, prediction, planning and action. Of these stages, prediction is the main focus of this thesis. Given a detailed map of the scene including road actors and infrastructure, the prediction module should understand the surrounding environment and anticipate how this will evolve, notably predicting the future movement of road actors. For example, if "A pedestrian is waiting at the crosswalk, gazing towards the road", there is a high chance that they are waiting to cross the road once it is safe, whereas if "A pedestrian is waiting at the the crosswalk with headphones on and looking at his/her mobile phone", this means that the pedestrian is not aware of the vehicles on the road and there is a high chance that he/she may start crossing the road without considering if it is safe." Similar abnormal behaviours can also be expected from minors such as "suddenly jumping from the footpath and running onto the zebra crossing instead of waiting until it is safe ". In the case of a vehicle, an observation that "A vehicle is travelling on a multi-lane highway with its ego lane occupied and the adjacent

overtaking lane empty with a positive relative speed with respect to the forward vehicle" makes it likely that the vehicle will perform a lane change manoeuvre to overtake the forward slow moving vehicle. Alternatively, if "a vehicle is slowing down near a junction and moving towards the outside lane", it is likely the vehicle will perform a turn manoeuvre at the junction. The prediction module should not only predict the future motion semantics but also estimate the future trajectories of other vehicles as any specific manoeuvre can be performed with different control of the drive and steering functions leading to different trajectories.

Once the future movements of all the other road actors are predicted, planning the future path for the ego vehicle is simpler, safer and comfortable. Since the ego vehicle already has a prediction of where the other road actors will be in the near future, it can avoid them using advanced obstacle avoidance techniques while planning its own path. In addition, this advance knowledge of how the scene will evolve makes it possible to reduce the requirement for emergency braking or acceleration, thus making the planned trajectory much smoother. This should make the journey more comfortable and reduce fuel consumption.

1.1 Problem Definition

The problem addressed in this thesis is how to predict the future movement of other surrounding vehicles in the vicinity. This prediction includes motion semantics and the future trajectories. The prediction should correspond with observations of the behaviour of human drivers, based on their previous experience, training and external factors. Prediction is difficult as human behaviour is very diverse and there is no obvious, universal model.

This is a well researched topic, but in most of the previous work, only multi lane highways and signalled intersections have been considered. In these cases traffic infrastructures play a very important role, biasing the prediction of future trajectories. For example at any traffic light intersection the movement of all the vehicles is heavily dependent on the traffic light phase which makes it harder to understand and analyze the impact of surrounding vehicle interaction on future movement. In this work, a newly collected dataset at two different un-signalled intersections has also been used along with the publicly available benchmark datasets to evaluate the developed models. The test vehicle used to collect this new dataset and bird's-eye views of both the intersections are shown in Figures 1.1 and 1.2 respectively.

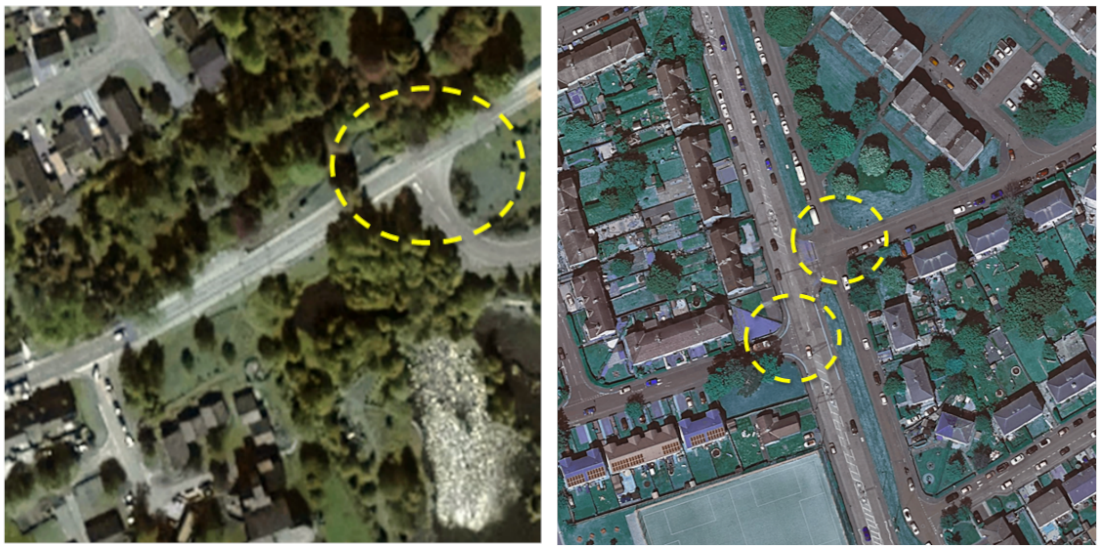
Moreover, in most of the previous research the vehicle trajectory prediction has only been performed for the next 5 secs from the current time instance. In some cases, for example, an overtaking manoeuvre on a single or dual carriage way, or a long deceleration on a major road approaching a turn onto a minor road, predicting

the future behaviour for only 5 secs might not be enough to plan the autonomous ego vehicle's future path safely. For long term prediction it is crucial to keep the scene information up to date for the entire prediction horizon. To address this, it is necessary to use the recently predicted positions of all the vehicles when predicting the next frame. This is absent in most recently developed models for behaviour prediction that rely only on the observed vehicle movements in past frames.

1.2 State of the Art

Vehicle behaviour prediction is well researched. Initially, predcitions were based on kinematic models such as constant velocity (CV), constant acceleration (CA), constant turn rate and acceleration (CTRA) etc. All these models are both efficient and accurate but only during the short term. Due to the lack of context, notably the relative position and velocity of the surrounding vehicles, these are inaccurate in the longer term. Consequently, there was a move towards interaction based models such as the Modified Social Force Model (MSFM) and Intelligent Driver Model to capture and utilize the effect of surrounding vehicles on the target vehicle's future behaviour. However, with the growing number of surrounding vehicles in any dense scenario, developing such complex interactive models can be very challenging. In order to overcome this, various learning based approaches were developed. Instead of creating hand crafted inter-vehicle interaction models, researchers tried to use recorded vehicle movement data to teach a model how a vehicle behaves in real world traffic. Initially only the target vehicle's past movements were used, but as stated above, behaviour is heavily dependent on the surrounding vehicles, so researchers started to use both these factors to improve the prediction accuracy. The basic work flow of these types of model is a two stage process. In the first stage both the target and surrounding vehicles' past movements are passed to the model to let it understand how the current scene has evolved during the past time frame; this is often called the observation sequence. In the next stage the model uses that understanding to predict how the target vehicle will behave in the future time frame; this is often called the prediction sequence. These types of models work well, but again better within the short term prediction horizon. This is because only the context of the observation sequence is used, and the future, short term updates are neglected in predicting the longer term movement. The entire prediction horizon is computed without using the knowledge of the how the scene may evolve in the near future.

In another paradigm, researchers have further sub divided behaviour in the prediction stage into a sequential, manoeuvre-trajectory process. First, a future manoeuvre such as left lane change, right lane change, straight on, left turn, right turn,



(a) Kingussie (T) junction (b) Edinburgh (four-way) junction

Figure 1.1: Bird's-eye view of two different unsignalized single-lane intersections where intersection specific vehicle trajectory data was collected by parking the test vehicle shown in Figure 1.2, at a safe locations near the junctions. The yellow dotted circles indicate the exact intersection locations.

is predicted, then this is used for future trajectory prediction. This makes it more aligned with a human driver's thinking process. First, a human driver decides which manoeuvre he/she will perform, then depending on the road geometry and context he/she plans the trajectory, e.g. at what speed and in what direction the manoeuvre will be executed. However, in this class of models the missing intermediate velocity prediction stage makes it difficult to understand and use the end-to-end vehicle kinematics while performing the final trajectory prediction.

Further, most of these developed models were evaluated either on a multi lane highway or at signalled intersections. In these areas the traffic infrastructure and highway codes play a very important role in determining the vehicles' future movement. This makes it harder to judge how well these models include proportionately the combined effects of the road layout and surrounding vehicles' behaviors on the target vehicle's future movement.

1.3 Thesis Objectives

The main objective of this thesis is to develop a behaviour prediction model which is effective not just in multi lane highways or signalled intersections but also at un-signalled intersections where the inter vehicle negotiations are to a much greater extent dependent on the movement of other road actors rather than the road infrastructure. The developed model should also be applicable to vehicle trajectories

Figure 1.2: Sensor equipped test vehicle used to collect data. A detailed schematic diagram of the test vehicle specifying where each of the sensors are installed, their specifications and end-to-end data collection pipeline is explained in Chapter 3 or Appendix.

collected from on-board sensors such as automotive radars, not just with infrastructure sensors such as cameras installed on top of high rise buildings or road network cameras. The model should also consider the limited range of vehicle sensors, and the likelihood of missed detections and occlusions.

In addition, future predictions should use the recently predicted, short-term positions of all the road actors in the vicinity to make use of the most relevant and recent scene information for long term prediction. The model must have the ability to capture the highly co-dependent spatio-temporal features simultaneously. Instead of directly producing the future trajectory, the model should first anticipate the planned manoeuvre of the target vehicle, then the likely velocity to predict the future trajectory.

1.4 Thesis Contributions

The main contributions of this thesis are as follows:

- A detailed analysis of the existing motion semantics and trajectory prediction algorithms, along with their drawbacks, is given in Chapter 2.
- A novel Convolutional Long Short-Term Memory (Conv-LSTM) architecture,

based on an Occupancy Grid Map (OGM) was developed which can capture the highly co-dependent spatial and temporal movements simultaneously. In contrast to existing techniques, the developed method predicts all the vehicles' movements for the next time-instance, and then uses these predicted positions to further predict the next to next and so on for each future time-instance to keep the scene information up to date.

- The above OGM based model was then changed and extended in a more efficient way. Instead of passing the high dimensional OGMs, only the target and surrounding vehicles' position with additional traffic rules and map based features, were passed to a conventional encoder-decoder architecture as feature vectors. This modification not only increased the training and testing efficiency by reducing the input data dimension, but also made it possible to provide the network with the aforementioned traffic rules and map based features.

- In contrast to the existing prediction techniques, the surrounding scene information was not only used in the encoder but also in the decoder to keep the scene information up to date during long term prediction. A dynamic surrounding vehicle selection technique was also employed to identify and select the most relevant surrounding vehicles instead of fixing them at the start of the observation sequence. Further, the model was extended with two additional intermediate tasks, manoeuvre and velocity prediction, followed by the future trajectory prediction, instead of only predicting the future manoeuvre before predicting the trajectory.

- Model evaluation was performed using publicly available benchmark datasets to compare with the state of the art, Then we used our newly collected radar dataset, e.g. Figure 1.1, to consider better the effects of sensing from a vehicle rather than road infrastructural sensors, and the considerably different road layout, a single carriageway and an absence of traffic signals and multiple lanes. For example, the four-way intersection shown in Figure 1.1(b) is slightly offset, chosen because it was possible to park the test vehicle (Figure 1.2) close to the junction to collect sufficient past movements of all the vehicles coming from all four directions before they performed their planned turn manoeuvre. During training this helped the model to understand how an individual vehicle's past velocity and relative position with respect to other vehicles influences the various turn manoeuvres.

1.5 Thesis Structure

- **Chapter 2:** A detailed literature survey of recently developed behaviour prediction techniques and how they can be used with various sensor modalities, such as stereo camera and automotive radar, is given. An in-depth explanation of some of the most commonly used deep learning techniques in the trajectory prediction area is also provided.

- **Chapter 3:** A novel methodology to capture the highly co-dependent spatio-temporal features simultaneously using a sequence of OGMs and a Conv-LSTM based architecture is given. Moreover, this chapter also presents a novel technique which can utilize the currently predicted positions of all the surrounding vehicles to keep the evolved scene more up to date during long term prediction.

- **Chapter 4:** A Long Short Term Memory (LSTM) encoder-decoder architecture is presented. This anticipates the future positions of other vehicles in the road network given several seconds of historical observations and associated map features. Unlike existing architectures, the proposed method incorporates and updates the surrounding vehicle information in both the encoder and decoder, making use of dynamically predicted new data for accurate prediction in longer time horizons. Experiments demonstrate that this approach can equal or surpass the state-of-the-art for long term trajectory prediction.

- **Chapter 5:** A summary of the strengths and limitations of the developed models, along with potential future work directions, is given in this chapter.

Chapter 2

Background and Literature Review

This thesis develops future behaviour prediction models to predict the future trajectory of all the surrounding vehicles. This is a crucial stage in any Advanced Driver Assistance System (ADAS) or in a level 5 autonomous vehicle in order to perform safe, collision-free navigation. This chapter reviews the key research advances in this area, together with how those techniques have been used with real-time sensor data.

Section 2.1 gives a brief history of autonomous vehicle development. Section 2.2 discusses existing deep learning tools, mainly focused on recurrent neural network models (RNN), as this is used heavily in my model development. Section 2.3 explains existing manoeuvre prediction, trajectory prediction and manoeuvre driven trajectory prediction models along with their shortcomings. Since automotive radar is the main sensor module considered in this work sections 2.4 and 2.5 explain its working principle and applications in the automotive sector respectively. Though we are interested primarily on data collected from the perspective of an automotive radar, similar techniques can be applied to data collected from other sensor modalities. Section 2.6 shows how similar features can also be extracted from the combined use of a stereo camera and a GPS/IMU toolkit.

2.1 Introduction

Remote controlled, then autonomous, self-driving cars were postulated soon after the invention of normal, human-driven motor vehicles. Francis Houdina demonstrated a radio-controlled car named “American Wonder” in 1925 which was able to drive through the congested streets of Manhattan without anyone sitting behind the steering wheel. The vehicle was able to start its engine, shift gears, accelerate, brake and also sound its horn when needed by the use of radio impulses sent by an

operator sitting in another car which followed this "American Wonder". Later, in December 1926, a Milwaukee (Wisconsin,USA) car distributor "Achen Motor" used this invented technology and exhibited its capability on the streets of Milwaukee and the surrounding territory under the "Phantom Auto" name [141]. This was demonstrated again on the streets of Fredericksburg (a city in Virginia) in June 1932.

Since then all the major automobile companies such as Audi, BMW, Ford, General Motors, the Google sibling Waymo [55], Nissan, Toyota, Tesla [148], [42], and Volkswagen have been developing advanced autonomous consumer vehicles [7] and not just light motor vehicles (LMV) but also heavy goods vehicles (HGV) or large good vehicles (LGV) [98] [165]. Various laws have been introduced and followed strictly by the automotive research community to keep the fatality rate to a minimum during the early stages [58].

To date, a significant amount of research has been done to successfully navigate an autonomous vehicle in multi-lane highways and signalled, multi-lane intersections. In all these cases the traffic infrastructures, lane markings and rules, traffic lights etc. play a crucial role. Navigating on a single carriageway or at a single-lane, unsignalled intersection makes the task a lot more challenging as there is minimal influence from the traffic infrastructure while negotiating with other vehicles. Even though different traffic codes already exist to handle these situations, such as the right of way, i.e. vehicles on the main road have higher priority, or that before performing a turn at any junction the driver must slow down and give a proper indicator signal, many human drivers fail to do this. Hence, to negotiate with other vehicles in these scenarios it is of paramount importance to not only follow the traffic codes but also infer the intention and future path of all other vehicles in the vicinity to avoid any potential collisions.

This chapter provides a detailed analysis of existing manoeuvre intention and future trajectory prediction techniques. manoeuvre intentions include both highway maneuvers (left lane change (LLC), right lane change (RLC) and straight-on) and intersection maneuvers (right turn, left turn and straight on). This work draws substantially on previous work on Recurrent Neural Networks (RNNs) to encode the sequence of input positions as observations and decode the future positions as prediction. These two RNN blocks are Long Short-Term Memory (LSTM) and Convolutional Long Short-Term Memory (Conv-LSTM) blocks where the LSTM takes sequential input in the form of a single dimension vector but the Conv-LSTM takes sequential input in the form of two dimensional image. The next section discusses the technical operating principles of these types of network.

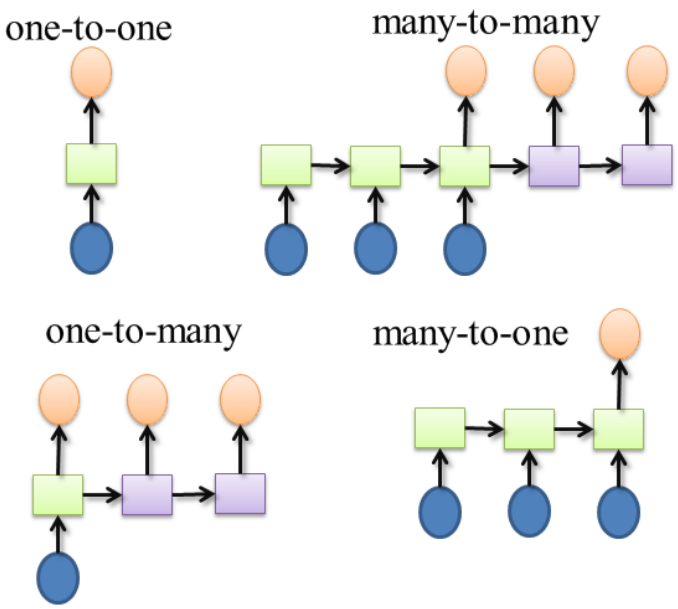


Figure 2.1: Four different types of RNN layouts, one-to-one, many-to-many, many-to-one, one-to-many, starting from top left and clockwise [99].

2.2 Deep Neural and Recurrent Neural Networks

The main focus of this thesis is to predict future trajectory as a sequence of positions given past observations, again a sequence of positions with some additional features. For this type of sequence-to-sequence problem, recurrent neural networks (RNN) have been used heavily by the community due to their capability to encode and decode any length of sequence [121], [142], [9], [76]. In order to understand the main contributions of this thesis, the following section provides a brief explanation about the working principles of recurrent neural networks.

2.2.1 Recurrent Neural Network Structures

A typical deep neural network has a single set of inputs and outputs only for the current time instance. In contrast, a recurrent neural network (RNN) has a sequence of inputs and outputs for consecutive time instances. It re-runs the same network at each time instance; this means that both the learned parameters, i.e. the network weights and biases, remain the same while maintaining and sharing additional "state" information across two consecutive time-steps. In this way, each RNN layer has two sets of inputs, i.e. the output and state information from the previous time-step and two sets of outputs, i.e. the outputs for the current time step and for the current state information which are used during the next time-step. There are four main types of RNN layout used commonly by the community. These are one-to-one, one-to-many, many-to-one and many-to-many [99] layouts, shown in Figure 2.1. The first, most basic RNN is the one-to-one. It can be thought of as

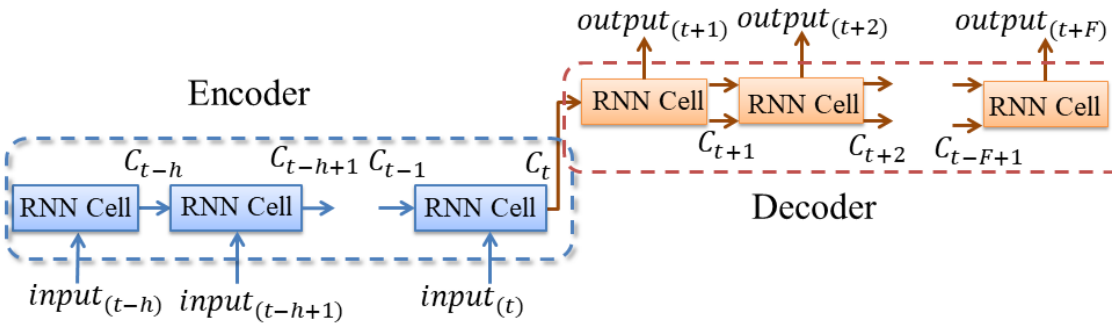


Figure 2.2: A typical encoder-decoder RNN structure where C_t is the state output from the RNN cell at time t, h is the observation sequence length and F is the prediction sequence length [164].

a typical DNN which takes only the current input and produces only the output for the current time instance. The second one-to-many layout takes one input data value for the current time instance and can produce an output sequence of any arbitrary length. A typical task for this type of RNN layout is to generate the caption of an image, a sequence of words of any length given a single image. The next layout is many-to-one which takes a sequence of inputs and produces a single output. This type of layouts has been used in various language processing tasks, e.g. classifying if a sentence is positive or negative in tone given the sequence of words, or manoeuvre classification tasks, e.g. identifying the current motion semantics (lane change or straight on) given the past sequence of positions (input trajectory). Finally the last many-to-many layout has both input and output sequences. Some typical examples include machine translation, where it takes a sequence of words (sentence) as its input and produces the translated version of the same sentence as a sequence of words, and trajectory prediction, where it takes the input trajectory as a sequence of positions and produces the future trajectory in the same form.

2.2.2 Encoder-Decoder Structures

The most commonly used many-to-many RNN layouts are called encoder-decoder models. Since both the inputs and outputs are sequences these are also called Seq2Seq [164] models. In these types of model the encoder takes the sequence input to encode the past observation state and the decoder produces the sequence output using the encoded state. Both the input and output sequences can be created using various data entities such as a sequence of words, i.e. a sentence [149], or a sequence of images, i.e. a video stream [19], or even across entities, for example in which the input sequence consists of images (i.e. a video) and the output sequence consists of words (i.e. a sentence) [162]. As shown in Figure 2.2, a typical encoder-decoder structure starts with random initialization of the current state of the RNN cell

in the encoder. It then takes a single time instance from the very start of the observation/input sequence. The RNN cell then produces a state output given the single time instance input and the randomly initialized state. The produced cell state is then used by the same RNN cell instead of the randomly initialized cell state at the beginning along with the next time instance input feature vector to produce the next cell state. This sequential cell state sharing technique continues until the end of the observation sequence. Once completed, the single encoder cell contains the encoded state of the entire observation sequence which is usually the current time instance. The next part is the decoder which also consists of a single RNN cell, but this time instead of randomly initializing the cell state, the encoded state is used as the initial state for the decoder RNN cell. At each time instance in the decoder the RNN cell produces two outputs, one the predicted entities and second the current state. Both these outputs are then fed back to the same RNN cell before predicting for the next time instance. Three types of RNN cell, i.e. the GRU, LSTM and Conv-LSTM cells, are discussed in the following section.

2.2.3 Recurrent Neural Network Cells

The training of a typical deep neural network consists of three stages, forward propagation, cost computation and backward propagation. Forward propagation is calculating the network output given a set of input data where the network weights are initialized randomly. In the next stage the difference between the current network outputs and their corresponding ground truth outputs are computed using a predefined cost function (aka loss function). Finally, the partial derivative of the cost function with respect to the current weights is computed to update the network weights. In the case of an RNN using conventional fully-connected layers, this may lead to a situation where the gradient of the cost function becomes extremely low. This will equally reduce the effective changes in the values of the network weights or can even make them zero in the worst case. This means the model will stop learning during the training stage. This the vanishing gradient problem. In order to overcome this issue three different types of RNN cells were proposed, which are the gated recurrent unit (GRU), long short-term memory (LSTM) and Convolutional long short-term memory (Conv-LSTM). All three types of cells are explained below:

The Gated Recurrent Unit (GRU)

The gated recurrent unit was first proposed by Kyunghyun Cho *et al* [22]. Compared to a classical DNN layer, a GRU cell has two additional gates, the"update gate" and the "reset gate". The update gate combines the current input and information passed from the previous time step using a weighted sum approach, which finally

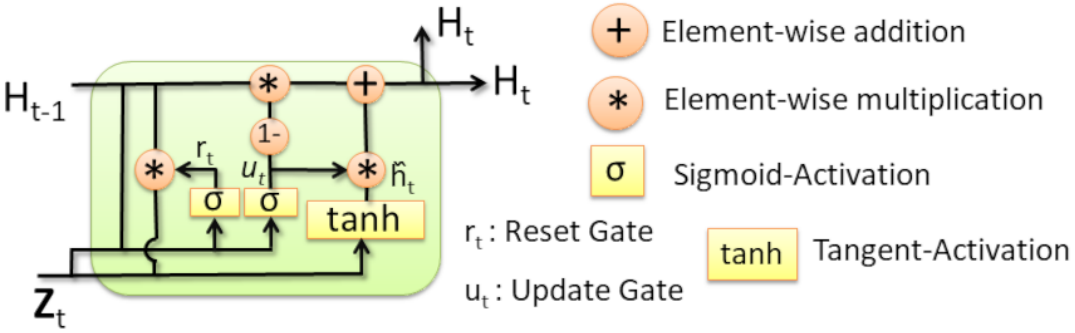


Figure 2.3: Gated recurrent unit (GRU) cell structure [22]

passes through a sigmoidal activation function to force it between 0 to 1. In this way the model decides how much past information should be passed to the future. The reset gate follows the same steps but to determine how much of the past information the model should forget. Finally the outputs from both these gates are combined to produce the final output for the current time step. A single GRU cell, shown in Figure 2.3, is formulated as,

$$u_t = \sigma(W^u * \mathbf{Z}_t + K^u * H_{t-1}) \tag{2.1}$$

$$r_t = \sigma(W^r * \mathbf{Z}_t + K^r * H_{t-1}) \tag{2.2}$$

$$\tilde{h}_t = \tanh(W^h * \mathbf{Z}_t + r_t * K^h * H_{t-1}) \tag{2.3}$$

$$H_t = (1 - u_t) * H_{t-1} + u_t * \tilde{h}_t \tag{2.4}$$

where $\mathbf{Z}_t$ is the input feature at time t, W^u, K^u are the update weights, W^r, K^r are the reset weights, W^h, K^h are the previous state weights, $+$ and $*$ denotes element-wise addition and multiplication respectively.

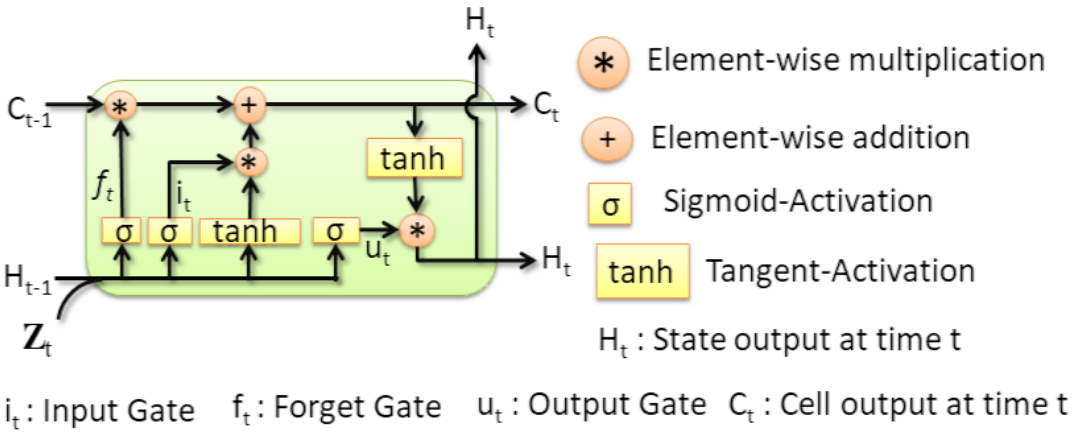


Figure 2.4: Long-Short Term Memory (LSTM) Cell structure [70]

Long Short-Term Memory (LSTM)

The long short-term memory (LSTM) cell was first proposed by S. Hochreiter and J. Schmidhuber [70]. The LSTM unit is an advanced version of a GRU unit with three different gates, "input", "output" and "forget" gates. In comparison to a standard GRU unit, an LSTM unit exposes both the current cell state (C_t) and output H_t at each time step which are then used at the next time step. Due to the additional "output" gate an LSTM cell works better with longer sequences when compared to the GRU. Given $\mathbf{Z}_t$, an input vector at time t, a single LSTM cell, shown in Figure. 2.4, is formulated as,

$$i_t = \sigma(W_{oi} * \mathbf{Z}_t + W_{hi} * H_{t-1} + b_i) \tag{2.5}$$

$$f_t = \sigma(W_{of} * \mathbf{Z}_t + W_{hf} * H_{t-1} + b_f) \tag{2.6}$$

$$u_t = \sigma(W_{oy} * \mathbf{Z}_t + W_{hy} * H_{t-1} + b_y) \tag{2.7}$$

$$H_t = u_t * \tanh(C_t) \tag{2.8}$$

$$C_t = f_t * C_{t-1} + i_t * \tanh(W_{oc} * \mathbf{Z}_t + W_{hc} * H_{t-1} + b_c) \tag{2.9}$$

where W_{oi}, W_{of}, W_{oy} and W_{oc} are the input weights, W_{hi}, W_{hf}, W_{hy}, W_{hc} are the previous state weights, b_i, b_f, b_y, b_c are biases, $+$ and $*$ denote element-wise addition and multiplication respectively.

Convolutional Long Short Term Memory (Conv-LSTM)

The convolutional long short-term memory (Conv-LSTM) cell was first proposed by X. Shi *et al* [179]. As shown in Figure 2.5, a Conv-LSTM layer works in a similar fashion to that of a Vanilla-LSTM (V-LSTM) except the inner representations and input are both two-dimensional (provided the input image is a single channel). This enables the model to capture both temporal and spatial correlations at the same time. This layer can be further formulated as,

$$i_t = \sigma(W_{oi} * OGM_t + W_{hi} * H_{t-1} + b_i) \tag{2.10}$$

$$f_t = \sigma(W_{of} * OGM_t + W_{hf} * H_{t-1} + b_f) \tag{2.11}$$

$$u_t = \sigma(W_{oy} * OGM_t + W_{hy} * H_{t-1} + b_y) \tag{2.12}$$

$$H_t = u_t \circ \tanh(C_t) \tag{2.13}$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_{oc} * \mathbf{O}_t + W_{hc} * H_{t-1} + b_c) \tag{2.14}$$

where OGM_t is the input 2D image at time t, an Occupancy Grid Map (OGM) for example, W_{oi}, W_{of}, W_{oy} and W_{oc} are input weights, W_{hi}, W_{hf}, W_{hy}, W_{hc} are previous state weights, b_i, b_f, b_y, b_c are biases and $*$ denotes convolution.

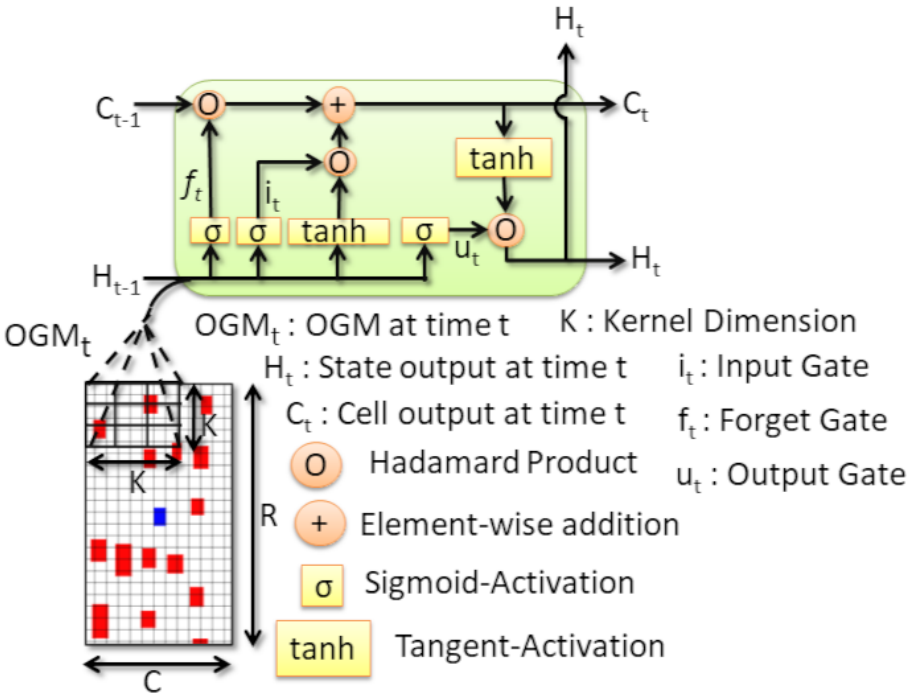


Figure 2.5: Convolutional Long-Short Term Memory (Conv-LSTM) Cell structure [179]

2.3 Trajectory and manoeuvre Prediction Methods

A detailed survey of vehicle behaviour prediction was published by Lefèvre [92], in which the models were classified into two main categories, physics-based and learning-based. Since then, a significant amount of work has been done using learning as a basis and state-of-the-art Recurrent Neural Network (RNN) architectures. The representation of any vehicle trajectory as a sequence of points/positions allows researchers to formulate the trajectory prediction task as an sequence-to-sequence (encoder-decoder [22]) problem, where a sequence of vehicle positions is fed into the encoder, and the vehicle's future positions are predicted recursively through the decoder. In considering further the previous work in this area, it can be further categorized into three sub categories, maneuver-prediction, trajectory-prediction and maneuver-dependent trajectory prediction.

2.3.1 Maneuver Prediction

The estimation of a vehicle's current manoeuvre is strongly correlated with given its past observation sequence in highways [178], urban intersections [123] and roundabouts [190]. In earlier research, classical approaches such as Support Vector Machines (SVMs) [106], [90] and Dynamic Bayesian Networks (DBNs) [93] were developed to classify a driver's intended manoeuvre 2-3 secs in advance. Considering a

Figure 2.6: A typical example of a survey performed to understand the importance of video sequence instead of a single image frame to judge a vehicle's motion and movement direction [91].

Markov process, some researchers tried initially to perform manoeuvre prediction using only the current state. Later, to investigate the suitability of this concept a survey was performed [91] in which one group of drivers was shown a single image frame, and the other group shown a video sequence, of several random traffic scenarios. Both groups were asked to anticipate future vehicle maneuvers. The group having access to the video sequence performed significantly better, which justifies the incorporation of vehicle's past states in addition to the current state during behaviour prediction. A typical example from this survey is shown in Figure 2.6, where if only the first image is seen, it is impossible to judge if both the vehicles i.e. A and B, are moving or parked. However, if three consecutive frames are seen, then it can be easily judged that vehicle B is moving and vehicle A is parked (static).

Moreover, interactions with and between surrounding vehicles and map based features also play a very crucial role during target vehicle's future manoeuvre prediction. This is why researchers started to incorporate both these factors into their multi-layered artificial neural networks for the following cases, lane change prediction on a highway [178] and turn prediction at an intersection [189].

In most cases, manoeuvre prediction was treated as a classification task, but in [181] a lane change manoeuvre was considered as a regression task with two major parameters, first, the time at which a driver begins to shift laterally within a lane and second, when the lane change will be completed (see Figure 2.7). In all these tech-

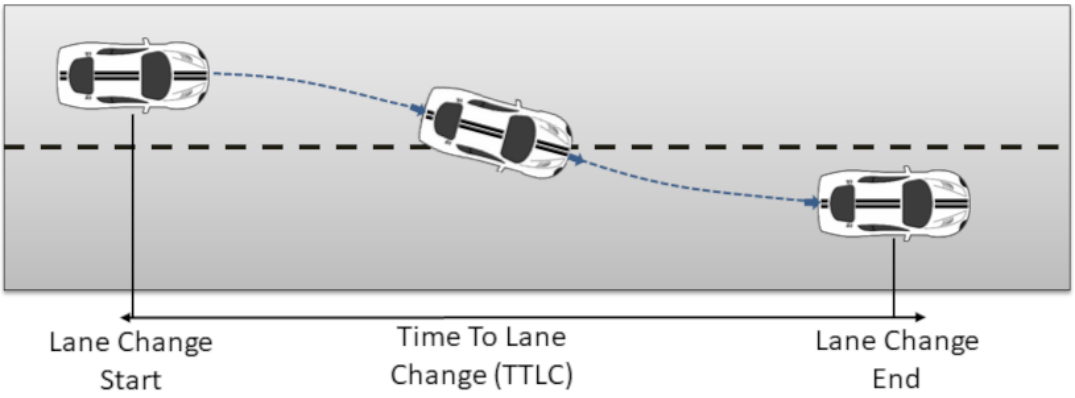


Figure 2.7: Lane change manoeuvre prediction treated as a regression problem using two parameters, lane change start and lane change end time [181].

niques only the future maneuvers of the surrounding vehicles were estimated. This is not sufficient to plan a collision free trajectory for the autonomous ego-vehicle as any specific manoeuvre can be performed at any particular velocity/motion leading to different positions in the next 3-5 secs.

Assuming future movement can be seen as a set of discrete decisions made by the human driver followed by a continuous movement, an elegant way to handle both discrete and continuous variables is the Hybrid State Model [50]. Such a model was used by Geng *et. al.* [54] to encode a strong implementation of the Highway Code, which defines rules which all traffic participants should (but don't always) follow. Most behaviour prediction models developed to date are scenario specific, e.g. designed for motorways or urban settings. This makes the problems tractable, and if taken to conclusion the road network would be split into situation specific nodes, and as such would require switching between behaviour prediction models [13]. Hence, to achieve full autonomy, a choice has to be made between two options that are difficult to scale, a scenario specific model that requires switching, and the dangers of delay and incorrect transitions, and a fully generic model that may be too complex for practical implementation.

2.3.2 Trajectory Prediction

Trajectory prediction has long been performed with classical Kalman [21], [161] and Particle [177], [90] Filters, Constant Turn Rate and Velocity models (CTRV) [124], or a combination of a cubic polynomial curve model with a CTRV [184]. They use only the target vehicle's (for which the prediction is being performed) past movement information to predict the future trajectory, which make them efficient and powerful in the short-term (i.e. 0.5s to 1s). However, these simple kinematic models are unable to perform well during the longer term (i.e. 3s to 5s in the future) as they do not encode either the interaction with other vehicles or with road

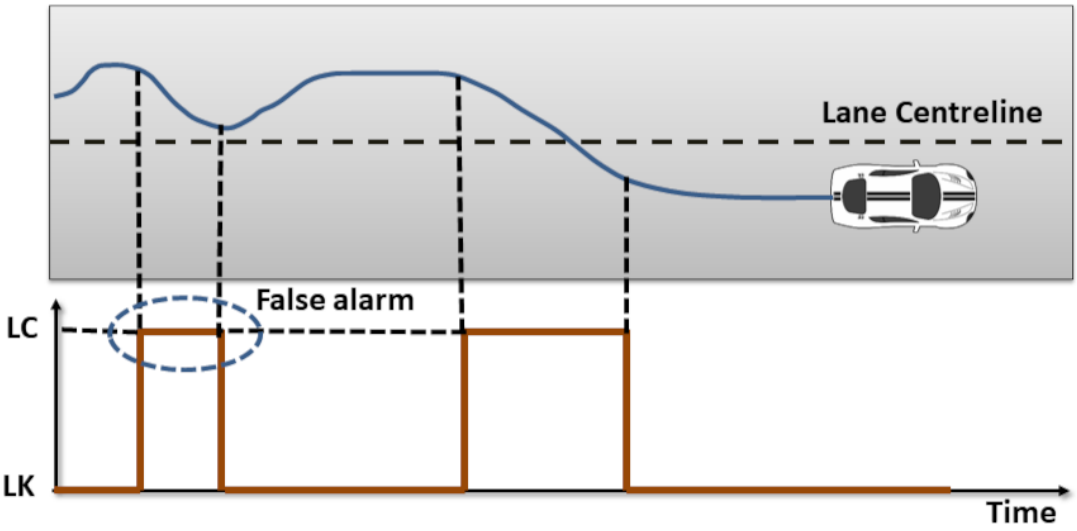


Figure 2.8: False lane change prediction from an SVM classifier due to zigzag movement of the target vehicle [174]. LC is lane change and LK is lane keeping.

infrastructure, which are essential for any long-term trajectory prediction.

Various interaction based models such as the Interacting Multiple Model (IMM) [152], Intelligent Driver Model (IDM) [96] and Modified Intelligent Driver Model (MIDM) [32] were developed to encode the vehicle to vehicle interactions, but all these models were mainly inspired by the classical car-following technique with a strong assumption that the target vehicle's future motion is solely dependent only on the single vehicle in front which is not always the case in any dense traffic or at an intersection. Inspired by the classical Social Force Model (SFM) [65], Ratsamee *et. al.* developed a Modified Social Force Model (MSFM) [127] where the underlying interactions were encoded not only through the physical components such as human position but also considered various human-centric social components such as face orientation, body pose, proxemics (personal space during motion) etc. Finally all these interaction forces with respect to the moving target were modeled using the MSFM to avoid any potential collisions and perform a successful navigation through a crowded scene. A similar approach, the n-body collision avoidance scheme was developed by Jur van den Berg *et al* [12] where each moving agent in the scene acquires the current positions and velocities of all the other agents along with its own position and velocity at each time step. Based on this information, an individual agent infers the permitted velocity for the next time instance to perform collision free movement. However, in the case of both MSFM and n-body collision avoidance, modelling these interacting force fields between individual agents and inferring future velocities based on other moving agents' velocities in a realistic fashion are very challenging. To avoid the development of complicated force-based or relative velocity-based models, various learning-based approaches using, for example, a Support Vector Machine (SVM) [174], [14], were developed. The target vehicle's past

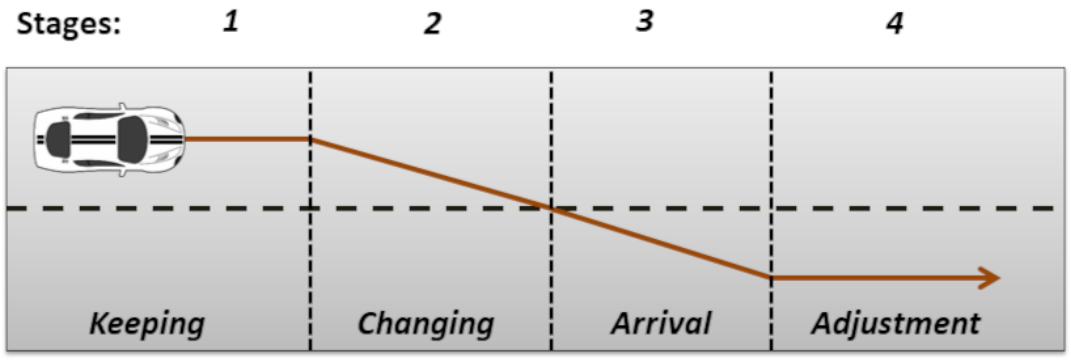


Figure 2.9: Four stages of a lane change maneuver, starting from lane keeping, lane changing, lane arrival and finally lane adjustment [174].

features like acceleration, heading, steering angle, lane position etc. were used to gain an early estimate of a lane change manoeuvre via future trajectory prediction or a Growing-Hidden Markov Model (G-HMM) [160], when only the past positions were used to estimate the future trajectory. Using Artificial Neural Networks (ANNs) [2], past observations were encoded through LSTM cells. In the case of the G-HMM [160], the model also had the capability to update both its structure and parameter incrementally whenever a new observation sequence became available. As shown in Figure 2.8, most of the lane change prediction models have a high false alarm rate during zigzag driving patterns. This zigzag vehicle trajectory mostly occurs when human drivers decide and start a lane change manoeuvre and then abort immediately after realizing the presence of surrounding vehicle in the adjacent lane. This problem was solved by first dividing the lane change manoeuvre into four stages, Lane Keeping, Lane Changing, Lane Arrival and finally Lane Adjustment [174] (see Figure 2.9), and then training individual SVM classifiers for each of these stages.

Most of these techniques incorporate the surrounding map information and also learn the model parameters from past observations, but the surrounding vehicle interaction information is still missing, and this plays a crucial role in future trajectory prediction. To incorporate these interacting effects the researchers started placing both the target and surrounding vehicles either in a single Occupancy Grid Map (OGM) [87], [121] or multiple Occupancy Grid Maps [114], divided on the basis of the number of lanes.

Recently, in ANN encoder-decoder architecture has attracted a lot of attention [121], [29], [115], [82] as a trajectory prediction problem can be considered as a sequence-to-sequence problem, where the both surrounding vehicle interaction information and map based features are added into the input sequence. In [121] and [29] the past position information was fed directly as an input trajectory sequence to predict the target vehicle's future position. In [115] and [82] a sequence of occupancy maps is provided to the model which is trained to predict the future map sequence.

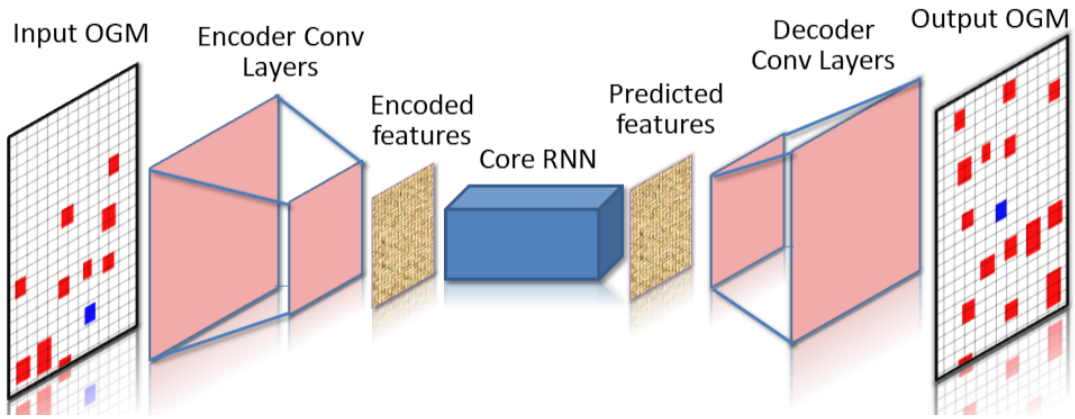


Figure 2.10: Occupancy grid map (OGM) prediction using multiple convolutional layers for spatial feature extraction and RNN cells (LSTM/GRU) for temporal feature extraction via classical encoder-decoder structure [115], [82].

An example of occupancy map input-output is shown in Figure 2.10, where the encoder consists of multiple convolutional layers to extract the spatial features from the input OGMs. The extracted features are then passed to the core RNN block to extract the temporal features. Using both the temporal and spatial features the core RNN block, mainly made of LSTM or GRU cells, performs the future motion prediction, but only in feature space. The decoder module consists of multiple transposed convolutional layers that take those predicted motions to produce the future occupancy map through up-sampling. In the first case, where the raw trajectory information was used, this makes it difficult to consider all the surrounding cars in the decoder as these only learn future movement from the encoded features. Therefore, the interaction predicted in the future horizon may not be fully exploited. In the second case, where a sequence of occupancy maps is provided, this can lead to two major problems, establishing car association and loss function design. First, associating each predicted car's position with its corresponding ground-truth position can be difficult when there are multiple cars close to each other. Therefore, the positions predicted by the network during training may be wrongly assigned, leading to a wrongly computed loss function. Second, the common loss function used in these cases is the pixel to pixel distance between the predicted and ground truth occupancy map which can be misleading in a scenario where more than one car is moving at the same speed.

In [114], instead of using the classical Recurrent Neural Network (RNN), a Relational Recurrent Neural Network (RRNN) was used. The advantage of the RRNN over the simple RNN is that it not only captures the inter-vehicle interaction in the observation sequence but also enables an "attention" mechanism to identify the most important information in the input and store this in the memory state accordingly. Inter-vehicle interaction effects can also be captured by connecting the underlying

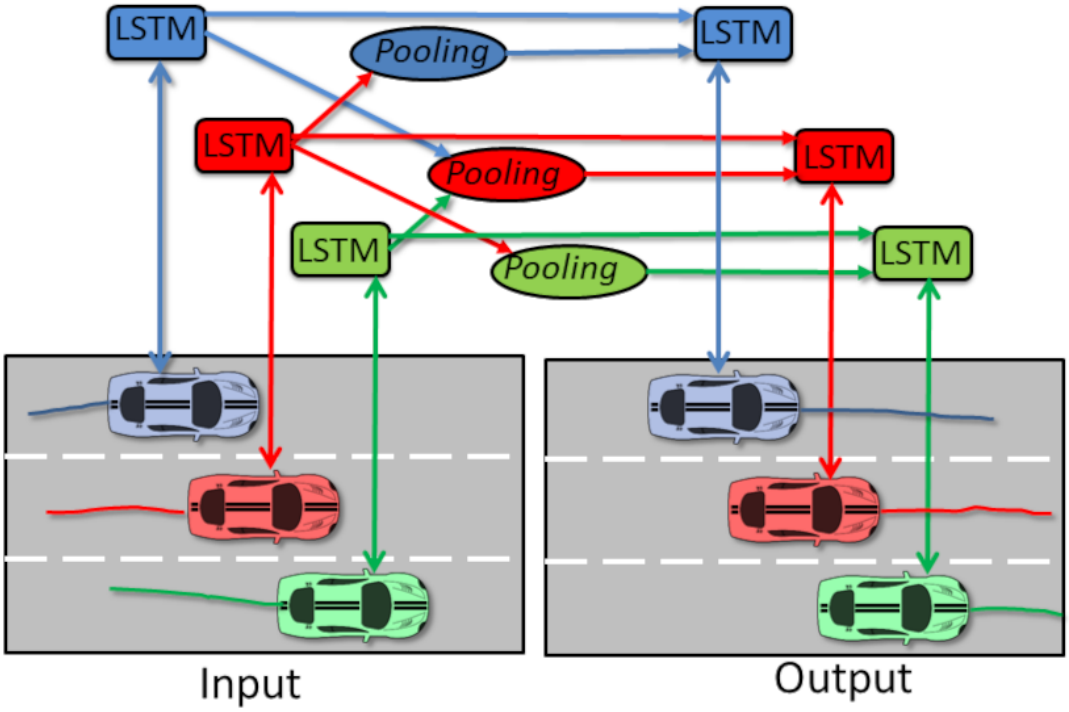


Figure 2.11: A typical example of a graph network where each LSTM cell captures an individual vehicle's temporal features including the target vehicle for which the prediction is being performed. The extracted features from individual LSTM cells are then shared through a pooling structure [63].

states of the LSTM cells, where a single LSTM is responsible for either the SV or the TV itself [63], [74]. This type of state sharing technique can be thought of as a graph- [100], [94] or a tree-network [119] where each traffic agent is considered as a node (a single LSTM Cell) connected to all other nodes via edges.

A typical example of a graph network is shown in Figure 2.11, in which the hidden states of each vehicle-specific LSTM cell capture the effective temporal information for that specific vehicle. It includes both the positions and motion intentions at different times. Lastly, a pooling structure was developed [63] which helps the target vehicle (for which the prediction is being performed) LSTM unit to accept and merge information from the neighbouring LSTM units. In a slightly different approach, along with individual LSTM cells, an additional set of Fully-Connected (FC) layers were added for each surrounding vehicle [44]. These additional FC layers were fed directly with the relative position between the corresponding surrounding vehicle and the target vehicle. This combined LSTM-FC network structure was called a Data Fusion encoder. Moreover, instead of using a single LSTM block in the decoder, four different LSTM blocks were maintained, in which three blocks were responsible for producing three different manoeuvre specific trajectories, left lane change, right lane change and straight on, and the fourth block produces three weights to combine the all three predicted trajectories into a single trajectory.

Inspired by a conversational model called DialogWAE (Wasserstein auto-encoder) [62] C. Fei *et. al.* proposed a conditional Wasserstein auto-encoder trajectory prediction model (TrajCWAE) [45] where the prior and posterior distributions were modeled as Gaussian mixture distributions and were minimized using Wasserstein-GAN (Generative Adversarial Networks) [4] instead of the classical KL-divergence [15] technique. In addition, kinematic physical constrains were added to the model to penalize the generated trajectories with large jerk and high acceleration as these are not practical in any real world scenario. In [175], instead of feeding the vehicle past trajectory information directly into the LSTM cells, G. Xie *et. al.* used an additional 1D convolutional layer followed by a 1D pooling layer to extract firstly the key features from the raw trajectory input and then feed the high level extracted features into the LSTM cell structure to sequentially predict the target vehicle's future positions. In addition, a box plot method was adopted to remove the outliers from both the training and testing trajectory data to improve the accuracy. The main purpose behind using the two-stage CNN-LSTM technique is that the CNN part will learn the spatial relations and the LSTM part will learn the temporal relations between the input and output trajectories of the target vehicles. However, this multi-stage training technique can make it harder for the model to learn jointly the highly co-dependent spatio-temporal features.

In any congested traffic scene a vehicle's movement is strongly dependant on the movement of the "front-vehicle" which further depends on the movement of the vehicle in front of the "front-vehicle". This chain of dependency can also go in the left and right directions, as far as the range of the latest on-board sensors, or even further with the help of infrastructure sensors. The problem with graph-networks is that they become very complicated with an increasing number of surrounding traffic agents. This forces users to consider only a few surrounding vehicles with respect to the target vehicle [74], making the scene information partial for the "front-vehicle". Moreover assignment of specific nodes [100] or LSTM cells [74] to the neighbouring traffic agents at the beginning of the observation sequence does not allow any new vehicle to appear in the vicinity, or any existing vehicles to disappear for the entire observation and prediction horizon ("birth" and "death" events). These events are very likely in any dynamic traffic scene such as at intersections or on motorways during high speed overtaking.

2.3.3 Maneuver Driven Trajectory Prediction

A driving process can be divided into two sub processes in which a human driver decides which manoeuvre (Left-Turn, Right-Turn, Lane-Change etc.) to perform and then plans the trajectory keeping the surrounding vehicles, road map and intended

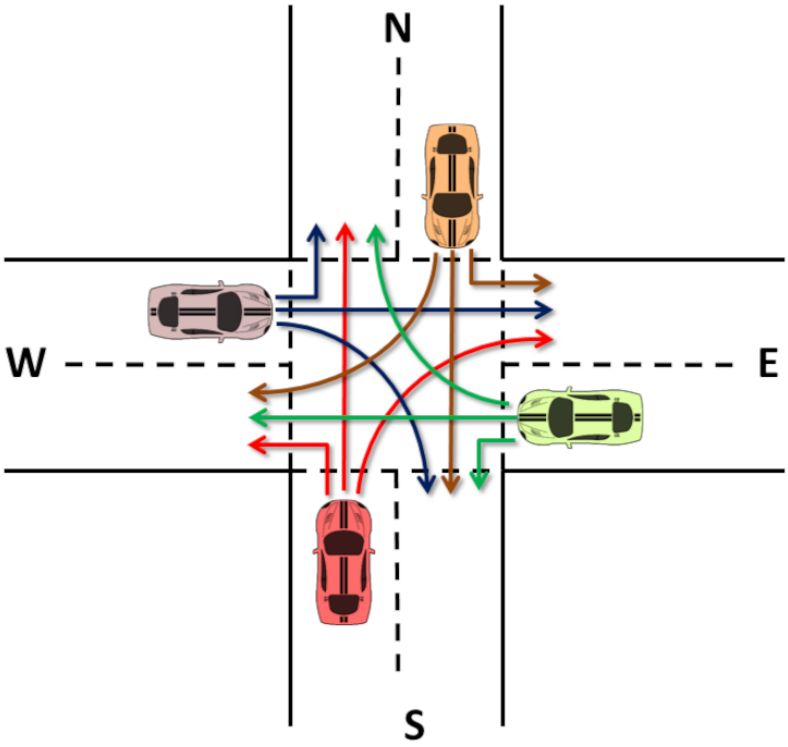


Figure 2.12: An example four-way intersection used in [145] to create the manoeuvre specific trajectory clusters.

manoeuvre in mind. Considering this inter-dependency between a vehicle's motion semantics and its future course, a lot of researchers have considered the trajectory prediction problem as a two stage task where the underlying manoeuvre intention aids prediction of the future trajectory or vice versa. Mohammad *et. al.* [145] used the past observation sequence to estimate the future trajectory first and then with the help of the Longest Common Subsequence (LCSS) distance metric identified which manoeuvre specific trajectory cluster was the closest match to predict the future maneuver. In order to create the trajectory clusters, vehicle data collection was performed at a typical four-way intersection (see Figure 2.12) using infrastructure sensors, mainly overhead CCTV cameras. The collected trajectories were then clustered around three common manoeuvre classes *i.e.* left turn, right turn, and straight on, with the help of K-means [97] and Expectation Maximization [116] algorithms. The problem with this approach is that the anticipated manoeuvre is not recursively helping the trajectory-prediction module, making the manoeuvre estimation stage redundant. Alternatively, given the observation sequence the underlying manoeuvre can be estimated first. This has been encoded using a Dynamic Bayesian Network [134], [135], [137], a Partially Observable Markov Decision Process (POMDP) [78], and a set of Hidden Markov Models (HMM) [49] where a single HMM model was associated with each manoeuvre and trained with the recorded maneuver-specific trajectory clusters or a single Hidden Markov Model integrated with a Gaussian

Mixture Model (GMM). Similar to multiple HMM models, a mixed regression approach was developed by Tran and Firl [153] where individual regression models were trained against each manoeuvre class to identify the most likely manoeuvre of the target vehicle given a new set of observation sequences. After successfully estimating the intended maneuver, the future trajectories were predicted either using the mean and variance of the trajectory cluster associated with the predicted manoeuvre [5], [6], or using Dynamic Time Warping (DTW) distances between the observed trajectory segment and individual trajectories within the predicted cluster to identify and use the best matching trajectory, together with the vehicle's current kinematics [131] or simply using the conventional Frenet Framework [17], [75]. In the last case, *i.e.* with the Frenet Framework, the authors assumed that during the entire process the vehicle's longitudinal acceleration remained unchanged and that all the vehicles travel along the lane centre line except for the times when it is actually performing the lane change maneuver. Even though in most cases vehicles do travel along the lane centre line, the assumption of constant longitudinal acceleration in [17] and [75] can make it hard for the model to understand the interacting effects of surrounding vehicles on the target vehicle's movement. Lane centre lines are usually extracted from a geographic information system (GIS) and stored in parabolic form shown in equation (2.15)

$$y(x) = c_2x^2 + c_1x + c_0 \tag{2.15}$$

where c_2, c_1 and c_0 are the coefficients. It is known from [170] that the lateral component $d(t)$ and longitudinal component $s(t)$ for a prototypical trajectory of a vehicle moving from an initial state $(s_0, \dot{s_0}, \ddot{s_0}, d_0, \dot{d_0}, \ddot{d_0})$ to the final state $(s_1, \dot{s_1}, \ddot{s_1}, d_1, \dot{d_1}, \ddot{d_1})$ in the *Frenet* frame can be optimally modelled as a polynomial of order 5. This guarantees the jerk continuity and provides a unique solution. Trajectory generation in the *Frenet* frame involves two major steps starting from estimating the initial as well as final state and then generating the trajectory.

Initial and final state of trajectories in *Frenet* frame The initial state trajectory of target vehicle shown in equation 2.16 was derived from its state vector which includes current position in the world co-ordinate frame (x, y), speed ν, orientation θ and acceleration ω.

$$\begin{cases} d_0 = y_0 \\ \dot{d_0} = \nu_0 \sin(\theta_0) \\ \ddot{d_0} = \omega_0 \sin(\theta_0) \\ s_0 = x_0 \\ \dot{s_0} = \nu_0 \cos(\theta_0) \\ \ddot{s_0} = \omega_0 \cos(\theta_0) \end{cases} \tag{2.16}$$

where (x_0, y_0), ν_0, θ_0 and ω_0 are the initial position, speed, orientation and acceleration, respectively. The partial knowledge about the final state is shown in equation (2.17).

$$\begin{cases} d_1 = y_t \\ \dot{d_1} = \nu_t \sin(\theta_t) \\ \ddot{d_1} = \omega_t \sin(\theta_t) \\ s_1 = x_t \\ \dot{s_1} = \nu_t \cos(\theta_t) \\ \ddot{s_1} = \omega_t \cos(\theta_t) \end{cases} \tag{2.17}$$

where (x_t, y_t), ν_t, θ_t and ω_t are the final position, speed, orientation and acceleration, respectively. The final location as well as the final orientation of the target vehicle are usually estimated using the predicted maneuver. For example, in case of a turn manoeuvre (x_t, y_t) will be (x_{link}, y_{link}) where x_{link}, y_{link} are the x-y coordinates of the future link's or lane's position to which target vehicle will move, estimated by the manoeuvre prediction model. Similarly for follow road, the same lane's centre line polynomial as shown in equation 2.15 was used, and for lane change the lane width was used as the lateral shift so that (x_t, y_t) can be estimated.

Trajectory Generation The lateral, $d(t)$, and longitudinal components, $s(t)$, of the generated trajectory in the *Frenet* frame are shown in equations 2.18 and 2.19 respectively,

$$d(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \tag{2.18}$$

$$s(t) = b_5 t^5 + b_4 t^4 + b_3 t^3 + b_2 t^2 + b_1 t + b_0 \tag{2.19}$$

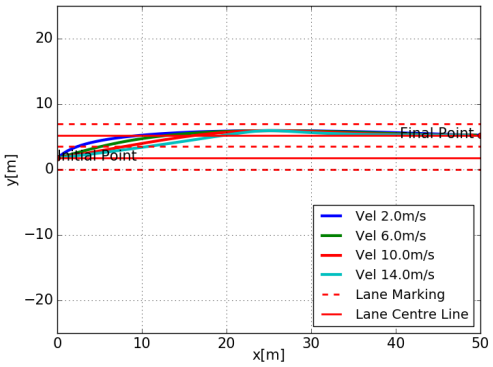
where $a_{i=0,1,2,3,4,5}$ and $b_{i=0,1,2,3,4,5}$ are the coefficients. With known initial and final states, starting time $t_0 = 0$ and ending time $t_1 = t$, equations 2.20 and 2.21 can be solved to estimate the coefficients which generate the predicted trajectory.

$$\begin{bmatrix} t_0^5 & t_0^4 & t_0^3 & t_0^2 & t_0^1 & 1 \\ 5t_0^4 & 4t_0^3 & 3t_0^2 & 2t_0^1 & 1 & 0 \\ 20t_0^3 & 12t_0^2 & 6t_0^1 & 2 & 0 & 0 \\ t_1^5 & t_1^4 & t_1^3 & t_1^2 & t_1^1 & 1 \\ 5t_1^4 & 4t_1^3 & 3t_1^2 & 2t_1^1 & 1 & 0 \\ 20t_1^3 & 12t_1^2 & 6t_1^1 & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} d_0 \\ \dot{d}_0 \\ \ddot{d}_0 \\ d_1 \\ \dot{d}_1 \\ \ddot{d}_1 \end{bmatrix} \tag{2.20}$$

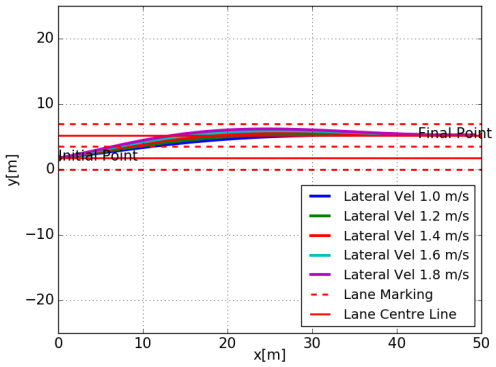
$$\begin{bmatrix} t_0^5 & t_0^4 & t_0^3 & t_0^2 & t_0^1 & 1 \\ 5t_0^4 & 4t_0^3 & 3t_0^2 & 2t_0^1 & 1 & 0 \\ 20t_0^3 & 12t_0^2 & 6t_0^1 & 2 & 0 & 0 \\ t_1^5 & t_1^4 & t_1^3 & t_1^2 & t_1^1 & 1 \\ 5t_1^4 & 4t_1^3 & 3t_1^2 & 2t_1^1 & 1 & 0 \\ 20t_1^3 & 12t_1^2 & 6t_1^1 & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} s_0 \\ \dot{s}_0 \\ \ddot{s}_0 \\ s_1 \\ \dot{s}_1 \\ \ddot{s}_1 \end{bmatrix} \tag{2.21}$$

A few examples of trajectories generated using the *Frenet* frame for turn and lane change maneuvers at different longitudinal and lateral motions are shown in Figure 2.13. A slightly different approach was adopted by Schlechtriemen *et al* [133] to perform the manoeuvre driven trajectory prediction task where the future manoeuvre probabilities were estimated by a Random Decision Forest and each manoeuvre specific trajectory cluster was derived using a Gaussian Mixture Regression method. Finally a "Mixture of Experts" approach was used to combine the computed manoeuvre probabilities and the probability density functions of the regression method to predict the future lateral movement of the target vehicle. However, road structure, e.g. the type of the junction, plays a crucial role during trajectory cluster formation. This makes these models intersection-specific and difficult to generalize.

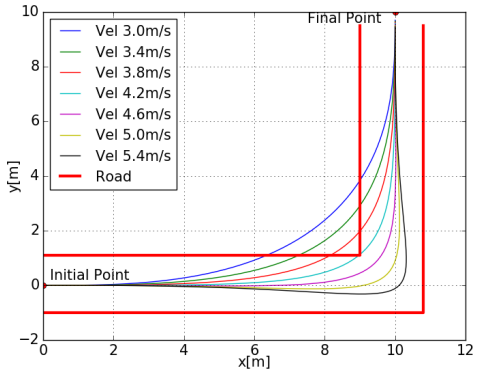
Using these two stage trajectory prediction concepts, various deep learning based models were developed for intention estimation and future motion prediction either simultaneously, e.g. with the help of Mixture Density Networks (MDN) [77], or separately, e.g. using two LSTM blocks where the first block predicts the future manoeuvre and the second block predicts the trajectory given the predicted manoeuvre from the first block [176]. In the last decade the LSTM encoder-decoder architecture has became more and more popular. This has the unique ability to read and generate sequences of any length, i.e. sequence-to-sequence. For example, a manoeuvre dependent encoder-decoder architecture was proposed by N. Deo and M. M. Trivedi [29] in which the future manoeuvre was predicated first given the observation sequence and then the decoder used the estimated manoeuvre along with the encoded state for recursive trajectory prediction for the entire future horizon. In another approach, inspired by one of the latest meta-learning induction [33]



(a) Left lane change with lateral speed and acceleration equal to $2m/s$ and $2m/s^2$ respectively



(b) Left lane change with longitudinal speed equal to 8 m/s.



(c) Generated trajectory for left turn.

Figure 2.13: Trajectory generation of target car in *Frenet* frame for turn and lane change maneuvers with different speed profiles.

approaches called Conditional Neural Process [52], Dong *et. al.* developed a novel two-stage network for sequential trajectory prediction [35]. The end-to-end network consists of an observer-subnet (or demonstration network) and a generator-subnet. The observer is responsible for categorization of the combined long-term past observation (past trajectories) of both the target and surrounding vehicles into different high-level manoeuvre oriented categories. The generator-subnet produces the future trajectory of the target vehicle sequentially, using only the immediate observation immediately before the prediction time step and the estimated category provided by the observer-subnet.

In all of these works, the future positions, both lateral and longitudinal, were directly estimated given the intended manoeuvre but the vehicle motion is affected firstly and as a consequence we see different trajectory splines. Forcing the model to predict the future motion first given the estimated maneuver, and then the future trajectory helps the network to understand the end-to-end maneuver-driven vehicle kinematics. Moreover a left-turn manoeuvre performed on different sides of a single/dual carriageway will have different position co-ordinates, but similar motion components (mostly lateral), which will help the model to generalize a single manoeuvre behaviour independent of the side from which the vehicle is approaching a junction. Further, in most cases, the models the models described in this section used the surrounding vehicles' (SV) interaction only in the encoder, and decoded the future trajectory of the target vehicle (TV) for the entire future horizon without using any updated information on the SV. This absence of updated scene information in the decoder makes the predicted trajectory inaccurate at longer horizons. Hence, inspired by [176] and [74], a maneuver-orientated trajectory prediction scheme was developed where the entire scenario was updated after every decoder prediction, considering each vehicle as the target vehicle concurrently, and identifying the most influential surrounding vehicles with respect to each target vehicle. This updates the latest scene input to the decoder, as opposed to the use of predetermined target and surrounding vehicle data [74].

2.4 Radar Sensing

A basic radar system detects the distance of an object by sending electromagnetic pulse and then measuring its time-of-flight using equation 2.22

$$r = \frac{C\tau}{2} \tag{2.22}$$

where C is the speed of light, τ is the round trip time and r is the distance in metres. A basic scheme of radar system is shown in Figure 2.14. Due to various

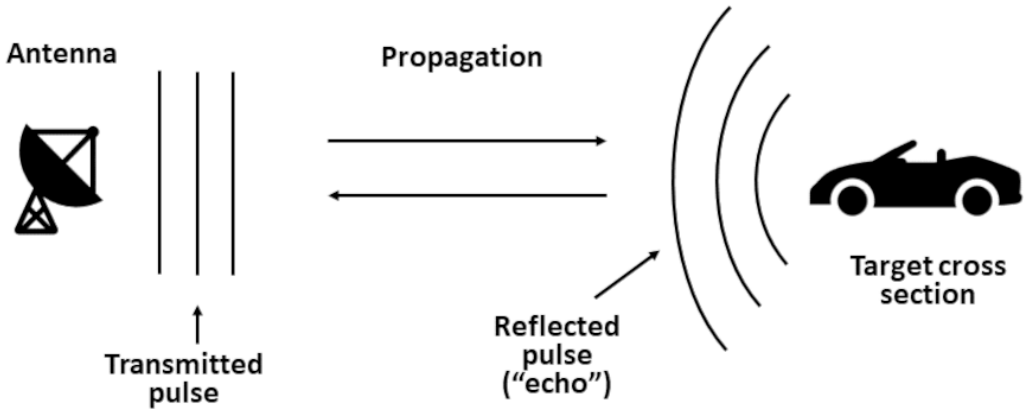


Figure 2.14: Basic scheme of a radar system [146]

atmospheric factors the return signal usually gets attenuated and loses its power over distance. Depending on the object's range from the radar system, its return signal power can be physically modeled by equation 2.23 [146]

$$P_r = \frac{P_t G_t A_r \sigma F^4}{(4\pi)^2 R_t^2 R_r^2} \tag{2.23}$$

where P_r is the return power, R_t is the distance between the object and the transmitter, R_r is the distance between the object and the receiver, G_t is the antenna gain, σ is the radar cross section, P_t is the transmitted power, A_r is the receiver antenna area, and F is the propagation factor.

2.4.1 Synthetic Aperture Radar

The radar system used in this thesis for test vehicle setup and data collection is an imaging radar. The most commonly used imaging radar technique is based on Synthetic Aperture Radar (SAR) technique. SAR images are usually created by transmitting successive pulses of radio waves from a moving target travelling at a high altitude like airplanes or spacecrafts. The reflected pulses from the ground plane are then captured and recorded using a receiver antenna. Finally, with the help of advance signal processing techniques such as back-projection algorithms high resolution ground images are constructed. An example SAR image and the way it was captured by moving the sensor is shown in Figure 2.15. Inverse SAR (ISAR) is another similar technique to SAR where a static radar sensor captures multiple moving targets. With the knowledge of the targets' movement and advance reconstruction algorithms high resolution images are created [173].

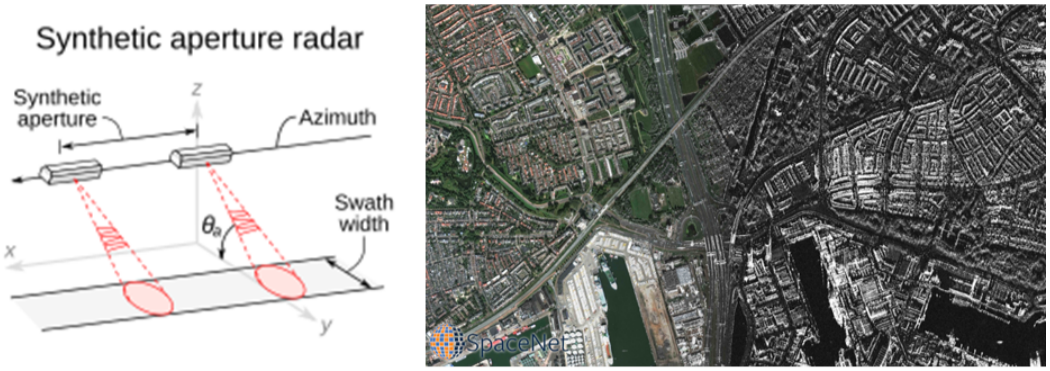


Figure 2.15: SAR image capture. An example of an optical (left panel) and a SAR image (right panel) is shown on the right. [173]

2.4.2 FMCW Radar

Frequency modulated continuous wave (FMCW) is the most commonly used technique in automotive radar. A FMCW radar consists of two components which are RF Generator and Mixer. A detailed description of both these components are mentioned below.

- **RF Generator:** The primary task of the Radio Frequency (RF) Generator is to generate a waveform with a starting frequency f_c, duration T_c and bandwidth B. Figure 2.16 shows an example of a chirp generated by the transmitter. The generated wave form will then be transmitted to capture the range information and the bandwidth size indicates how much data it can carry, implying higher range resolution. The generated waveform can be defined mathematically as function of frequency and time:

$$y(t) = A\sin\left(2\pi f_0 t + \pi k t^2 + \theta\right) \tag{2.24}$$

  where $k = \frac{f_1 - f_0}{\tau_P}$, f_0 and f_1 are the initial and final frequencies respectively and τ_P is the time difference between them. The most commonly used modulation techniques are Triangular wave and Sawtooth wave.

- **Mixer:** The main task of mixer is to mix the transmitted and received signal in order to estimate the range information. Assume the transmitted and received signals are x_1 and x_2 with frequencies ω_1 and ω_2 and phase θ_1 and θ_2, respectively. Both these signals can be formulated as follows:

$$x_1 = \sin\left(\omega_1 t + \theta_1\right) \tag{2.25}$$

$$x_2 = \sin\left(\omega_2 t + \theta_2\right) \tag{2.26}$$

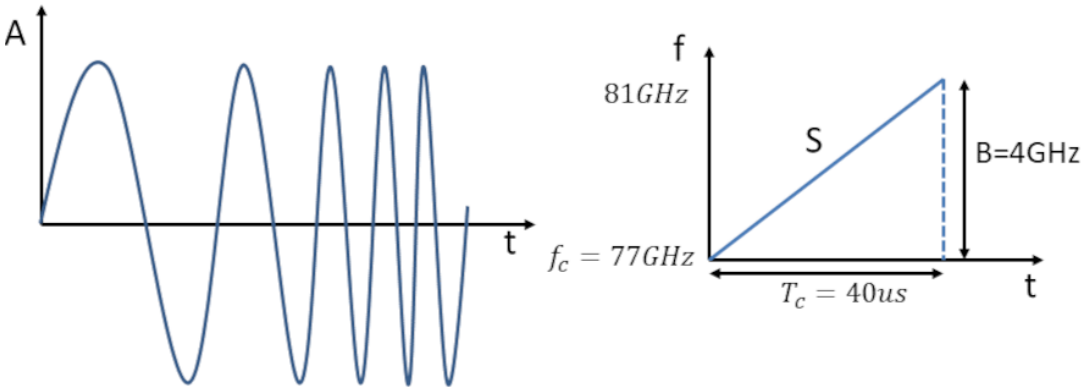


Figure 2.16: An example chirp generated by a FMCW automotive radar.

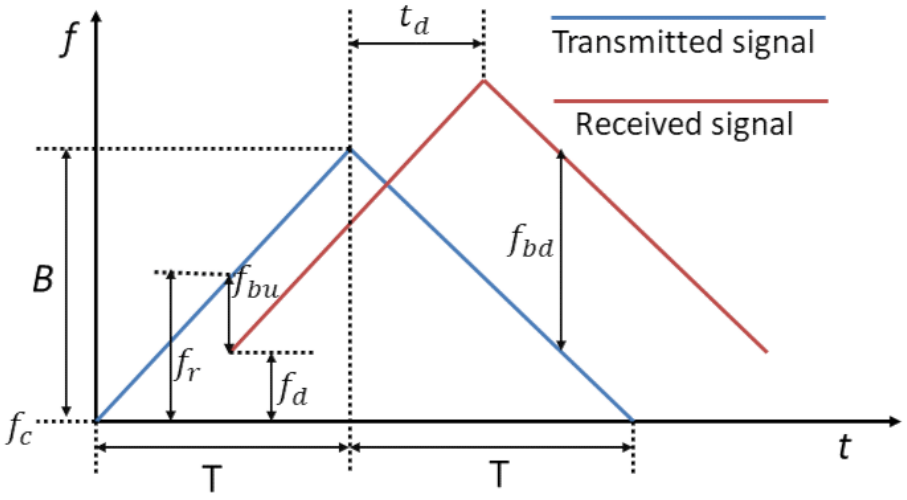


Figure 2.17: FMCW transmitted and received triangular chirps [172].

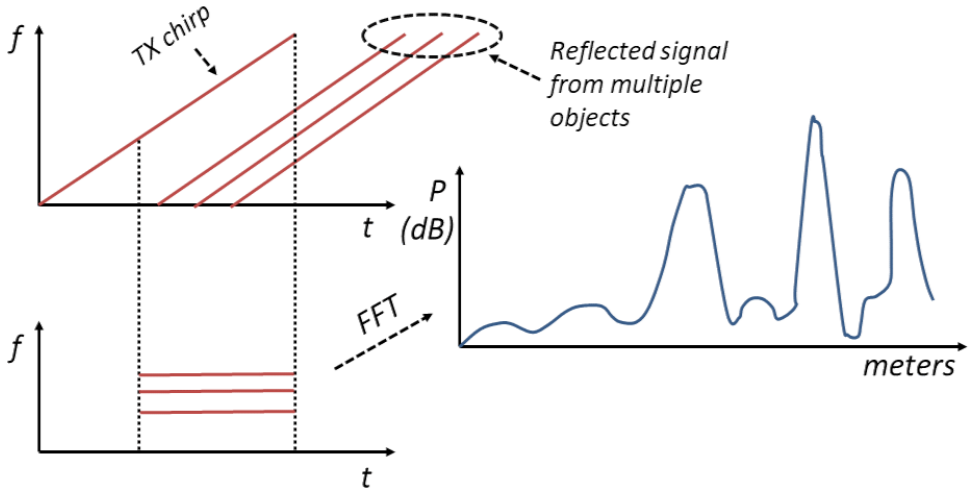


Figure 2.18: An illustration of how to transform a signal in the time domain to range data [79].

The mixer will first evaluate the phase and frequency differences of the above two signals. The resultant signal will be:

$$x_1 = \sin\left[(\omega_2 - \omega_1)t + (\theta_2 - \theta_1)\right] \tag{2.27}$$

In the next stage, the beat frequency up f_{bu} and down f_{bd} and the Doppler frequency f_d (see Figure 2.17) will be estimated using Fast Fourier Transform (FFT). Figure 2.18 demonstrates how a typical signal in time domain can be converted to range information through FFT. As shown in Figure 2.19, the range resolution of a radar solely depends on its bandwidth.

Most of the recent automotive radars are multiple input multiple output (MIMO) based, which means it will have multiple transmitter and receivers. With the help of multiple antennas the direction of arrival (DoA) can also be computed. The angular resolution of a MIMO radar can be computed using the following equation:

$$\theta_{res} = \frac{\lambda}{Ad\cos(\theta)} \tag{2.28}$$

where θ is the angle of where the object is, d is the distance and A is the number of antennas. Assuming θ is 0 and $d = \frac{\lambda}{2}$, the angular resolution will become:

$$\theta_{res} = \frac{2}{A} \tag{2.29}$$

For a MIMO radar with 3 transmitters and 4 receivers the angular resolution can be calculated using Equation 2.29:

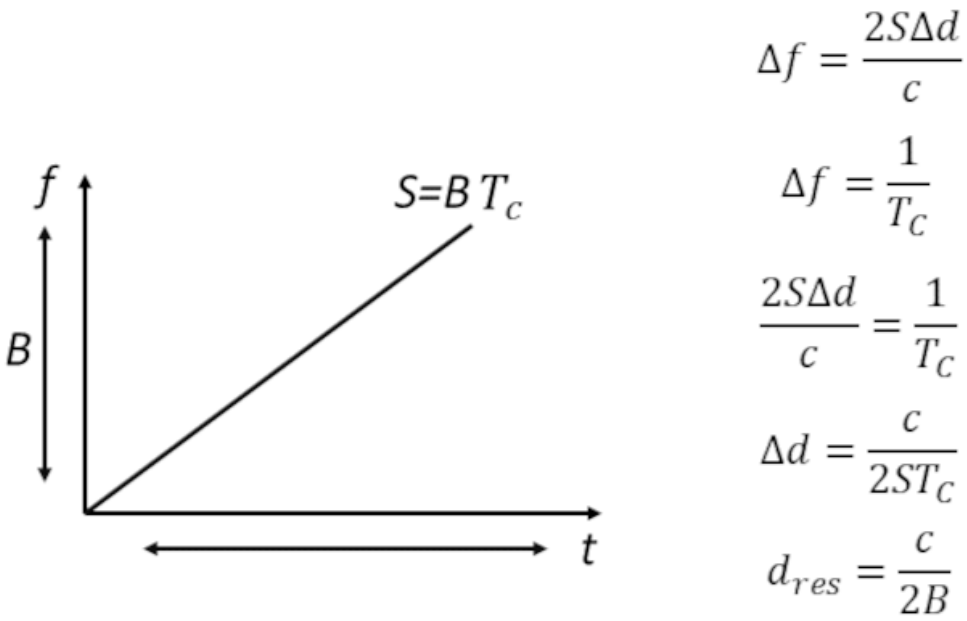


Figure 2.19: Derivation of the range resolution equation [79].

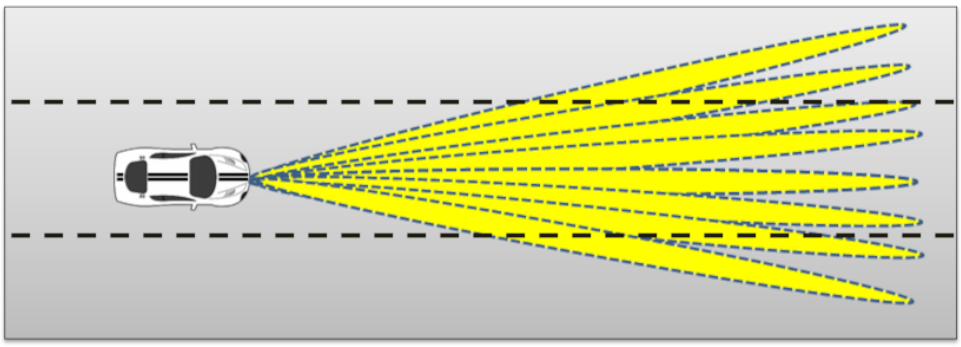


Figure 2.20: A typical example of a scanning radar installed at the front of an automotive [81].

$$\theta_{res} = \frac{2}{7} = 0.28rad = 16.043^{\circ} \tag{2.30}$$

An angular resolution of 16° is not always enough for high resolution imaging. With the help of mechanical scan it can be improved and the angular resolution will be based on the beamwidth. A typical example of a scanning radar with a range up to 100m is shown in Figure 2.20 .

2.5 Applications of Automotive Radar

Given the partial emphasis of this work on interpreting the data collected by an automotive radar system, this section describes some of the previous work in which such systems have been used. Applications of radar in the automotive sector were

initiated in early 70s [60], and since then radar has been used for many surround-sensing tasks due to its functionality, robustness, reliability and its ability to work in many adverse weather or lighting conditions [86], [147], [64]. An automotive radar consists of three building blocks. First, there is the sensing block for modulation, demodulation and raw data signal processing. Second, there is the perception block for target detection and tracking. Finally, the planning block involves localization, situation interpretation plus prediction and decision making [166]. Early automotive radar systems were really good at measuring object distance with high range accuracy but due to the poor angular resolution it was harder to separate two nearby objects, specially in case of small and vulnerable road-users such as pedestrians and bicycles. However, the newer generation of automotive radar has implemented various advanced and efficient signal processing techniques to achieve higher angular resolution and overcome this problem [43], [136]. The major applications have involved:

- Detection and tracking of other road users, e.g. vehicles [84], [112], [61], [105], pedestrians [20], [67] and bicyclists [66]. In the case of pedestrians and bicyclists the micro-Doppler signatures were used generated either from the leg movements of pedestrians or wheel rotations of bicycles.

- Target classification among four different classes (car, bicycle, single and group of people) using either a classical convolutional network, a residual network, or a combination of convolutional and recurrent networks [3].

- Human behaviour classification, e.g. slow walk, fast walk or slow walk with hands in pocket [140], together with classification uncertainty [39].

- Detection of lane and road markings [47], [46].

Mono-static radar systems have been used to detect and track targets using Kalman [102], cascaded Kalman filters [95], Extended Kalman Filters [150], polynomial fitting [104] and deep learning techniques [36], [122]. The use of multiple radars improves the accuracy [48]. Automotive radar based tracking starts by measuring three target parameters, range, radial velocity and azimuth angle [102], of which radial velocity will be used at first to separate the moving targets from the clutter. Multiple returns from the same target will then be grouped together to form target specific clusters which will then be associated with the existing tracks using the nearest neighbour approach. Finally the tracks belonging to the real targets are passed to a Kalman filter to estimate the current position of all the targets. Usually, target positions are converted firstly from polar to Cartesian co-ordinate system using the measured range and azimuth angle before feeding it to the Kalman filter.

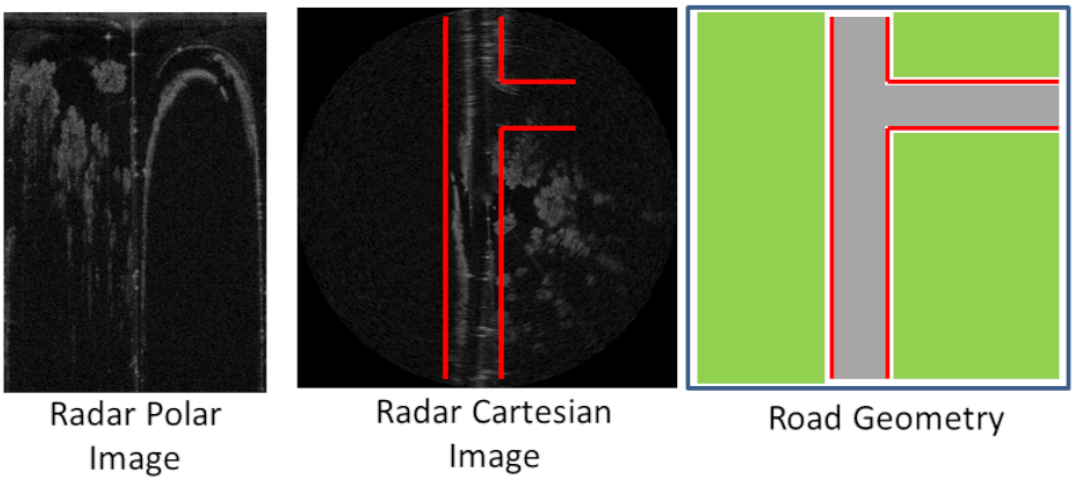


Figure 2.21: Advance Path Measurement technique [157], [156]. The left most image is the output from the Navtech radar in polar co-ordinates, then the converted image in Cartesian co-ordinates and finally the extracted road geometry.

Cao *et. al.* proposed an extended target tracking technique in [18] for vehicles with the assumption that the targets are rectangular. This prior shape information of the targets was added to the model as a quadratic equality constraint on the state to improve the target shape estimation plus it made the target association stage a lot more simplified and accurate.

Automotive radar systems are not now limited to specific target detection and tracking, but are also used to perform semantic scene segmentation, as for example by use of a two-fold neural network model [138]. Usually radar systems only work within the line-of-sight, but an understanding of temporal sequences of Doppler velocity and position measurements has allowed researchers to untangle the noisy indirect and direct reflections to make detection, classification and tracking possible even without line-of-sight [132].

Developers have used automotive radar to collect vehicle trajectory data [120] and to sense infrastructure sensor as a "birds-eye view" [158], [118]. Various radar based vehicle trajectory prediction and collision avoidance algorithms have been developed, but usually either for very basic lane changes on highways [120], [88], [101] or at specific intersections to predict a manoeuvre but not the entire trajectory [158]. Instead of using only radar, a Kalman filter based sensor fusion technique was adopted in [8] to fuse sensor data from LiDAR, Camera and Radar along with a vehicle-to-vehicle communication system to not only predict the surrounding road actors' future trajectory but also to anticipate both vehicle-to-vehicle as well as vehicle-to-pedestrian collisions well in advance. However, as shown in Figure A.2 in a foggy scenario fusing radar data with camera and LiDAR data can sometimes lead to poor perception of the scene. Millimeter wave automotive radars were also used by Tsang *et. al.*, to predict the ego-vehicle's future trajectory by first estimating

Figure 2.22: Test vehicle equipped with Velodyne LiDAR, Navtech Radar, ZED Stereo Camera and Advance Navigation GPS/IMU kit, used to collect data from different junction.

the road path ahead. The technique was called Advance Path Measurement (APM) [157], [156] which consists of four steps starting with first generating a 3D radar image of the current scene in polar co-ordinates using radar range, magnitude and azimuth data which are then converted into Cartesian co-ordinates. In the next stage, the road edge reflections were used in combination with a second order polynomial fit to define the path ahead of the ego-vehicle and finally its future trajectory, again with the assumption that a vehicle always travels through the road central line.

A example of data from the Navtech radar, converted from polar to Cartesian images, and the extracted road geometry, is shown in Figure 2.21. A similar technique was developed by Hoare *et. al.* [69] where the main focus was to estimate the detailed road geometry in front of the host vehicle using convolutional neural networks applied to the radar image. Moreover, radar was used jointly with other sensors, camera, LiDAR, and even a V2V communication system, to perform trajectory prediction and avoid any potential collisions. In bad weather or poor lighting a reliance on fusion of radar with other sensor modules can lead to a significant loss of performance, and a strong dependency on the V2V communication [187] can lead to dangerous accidents during dropped connections.

Hence, a vehicle detection and prediction system based solely on radar sensing is crucial for level 4 and level 5 autonomy in all conditions [34], and behaviour prediction is the main focus in this thesis. For the radar images, the test vehicle [143] shown in Figure 2.22 was used, which is equipped with a Navtech Radar, a Velodyne LiDAR, a ZED Stereo Camera and an Advanced Navigation GPS/IMU Tool Kit. The Navtech Radar used in this work operates at 76-77GHz with a maximum range of 50-200m (dependent on mode), a range resolution of 6-25cm, an azimuthal resolution of $1-2^o$ which equates to approximately 69cm at 20m and a maximum frame rate of 4Hz.

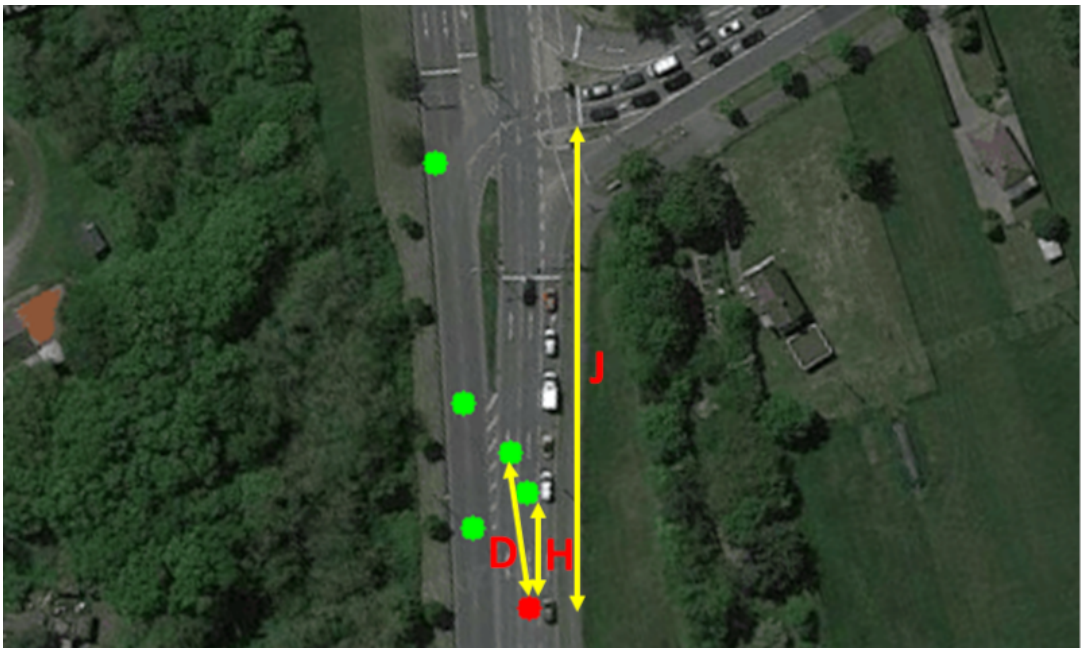
2.6 Feature Extraction Techniques

As mentioned earlier, most of the trajectory prediction techniques rely heavily on the current context which includes both the map based features extracted from a GIS and the relative positions of all the surrounding vehicles with respect to the target vehicle. This combined use of context information with raw measurement can boost the system performance significantly [51]. In some cases state-of-the-art semantic segmentation algorithms [113] have been applied to street scenes to classify image pixels into object categories. However, such detailed information may be redundant, as humans are selective and focus on salient details while driving safely. For example, patches of ice on a road are very important, but whether the pavement is composed of flagstones or tarmac is not an issue. Interactions of all other traffic participants with the ego-vehicle are key features need to be incorporated to make an accurate prediction [68]. For example, the detection of other cars in adjacent or in the same lane as the ego-vehicle make it possible to make a lane change prediction for a target vehicle [107]. The inclusion of prior information together with the state vectors of individual traffic participants should make the system more accurate and capable of longer term behaviour prediction [51]. This includes road structure, map based localization, road markings, traffic signs and lights etc [126]. Scene specific conditions, e.g. fog, ice, traffic light changes etc can also affect the dynamic behaviour of other vehicles [128].

The major steps involved in any feature extraction process are detection, classification, tracking and finally placing them within the GIS coordinate system. Though this thesis is mainly focused on automotive radar, this section explains that these features can be extracted from RGB images (see Figure 2.23(a)) collected with a stereo-vision system. In order to map all traffic participants (cars in this case) one needs the relative horizontal and vertical distance of each car with respect to the ego vehicle in metres. Using the stereo vision system combined with a trained DNN to recognize cars, an example of the detection system is given in Figure 2.23(b), with the corresponding positions on the global map shown in Figure 2.23(c). It can be seen here that vehicles on both sides of the road have been considered. Arguably, in the case of a multi-lane highway, this is not necessary as vehicles cannot cross from one side to another but in the case of single carriageway or intersections there is a high chances of this occurring. On a single carriageway during an overtake manoeuvre a vehicle has to move to the opposite side of the road. In the case of an intersection, the vehicle has to consider any oncoming vehicle on the other side of the road near the intersection before crossing or preforming a different turn manoeuvre at the same time, thus causing a collision. For distance estimation the detected bounding boxes were first overlaid onto the depth image generated using the left and

(a) Ego-vehicle perspective RGB image collected with the on-board stereo camera [53].

(b) Detection and classification of traffic participants in the scene. The bounding boxes are the detected target cars in the scene and the numbers in red are distances of the target cars from the ego vehicle in metres.



(c) Localization of traffic participants on a Google satellite map [109]. The red dot is the ego vehicle and the green dots are the surrounding vehicles where each is considered as a target vehicle once during behaviour prediction. The individual distance vectors of each extracted feature are shown by yellow arrows. For simplification and better understanding all these features are shown with respect to the ego vehicle.

Figure 2.23: Detection, Classification and Localization on Kitti dataset [53].

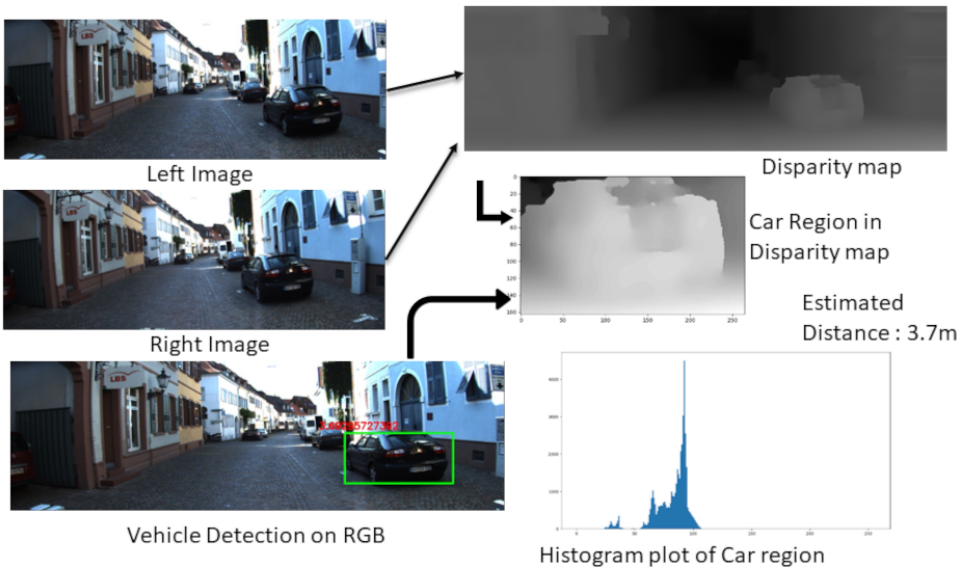


Figure 2.24: Steps involved in surrounding vehicle distance estimation using RGB stereo images from Kitti dataset [53].

right RGB images. Since each bounding box contains the detected vehicle as well as some background in the corners, a histogram based technique was used to estimate the longitudinal distance instead of simply averaging out all the pixels within the box. Since the detected vehicle and the background are at different distances, use of the histogram helped filter out the background pixels from the depth image. The detailed steps involved in this process are shown in Figure 2.24.

Next, we consider the impact of some of the most commonly used features such as headway distance in ego-lane H, distance from nearest car in adjacent lane D, speed of target car ν_T, relative speed from forward car ν_{diff}, distance from nearest junction J, indicator signal I and lane number or relative lane position L on behaviour prediction. The assignment of a target car T_{car}, car in an adjacent lane Adj_{car} and a forward car F_{car} with respect to the ego car are shown in Figure. 2.25.

1. *Headway distance in ego-lane H:* Distance of the T_{car} for which the prediction will be performed from the F_{car} in the same lane is the headway distance (see Figure 2.23(c)). This is one of the most important reasons for lane change. People tend to avoid changing lane as long as the ego lane is available or H is greater than the safety limit decided by highway code. On the other hand if the distance between T_{car} and F_{car} is low, people change lane to perform an overtaking manoeuvre based on the availability of next lane, or reduce their speed to increase the gap if the adjacent lane is not available. The headway distance H can be calculated by

$$H = \sqrt{(x_T - x_F)^2 + (y_T - y_F)^2} \tag{2.31}$$

where (x_T, y_T) and (x_F, y_F) are the locations of T_{car} and F_{car}, respectively.

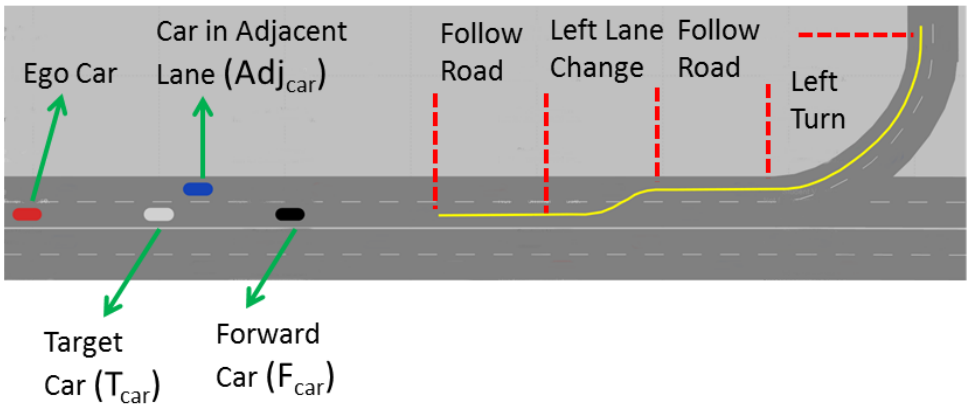


Figure 2.25: Assignment of T_{car}, F_{car} *and* Adj_{car} on a simulated scene along with possible locations for manoeuvre transitions (Follow Road, Left Lane Change and Left Turn shown with red lines).

2. *Distance of nearest car in adjacent lane D:* As mentioned above, the final decision before an overtake manoeuvre is dependent on the distance of nearest car in the adjacent lane, (which defines lane availability), which can be defined as

$$D = \sqrt{(x_T - x_{Adj})^2 + (y_T - y_{Adj})^2} \tag{2.32}$$

where (x_T, y_T) and (x_{Adj}, y_{Adj}) are the locations of T_{car} and Adj_{car}, respectively.

3. *Speed of target car ν_T:* Different manoeuvre sequences have likely speed profiles. For example, most drivers tend to reduce speed during a turn sequence. The follow road manoeuvre can have a wider speed profile based on the current context. The current speed of the target car, T_{car}, is a good starting point for future speed estimation since the acceleration and declaration are limited.

4. *Relative speed with respect to forward car ν_{diff}:* A high relative speed between T_{car} and F_{car} is another reason for an overtake maneuver. During this scenario the time to collision with F_{car} is low, which makes the driver in T_{car} either reduce speed or change lane based on availability of adjacent lane. ν_{diff} can be derived by

$$\nu_{diff} = \nu_T - \nu_F \tag{2.33}$$

where ν_T, ν_F are the velocity of the T_{car} and F_{car}, respectively

5. *Distance from nearest junction J:* The presence and type of a junction affects future speed and manoeuvre sequence significantly. If the distance of the nearest junction from T_{car} is high the chances of any turn manoeuvre in the near future will be low and vice-versa. J is defined as

$$J = \sqrt{(x_T - x_J)^2 + (y_T - y_J)^2} \tag{2.34}$$

Figure 2.26: Car indicator signal extraction. Image on left is input image to the pipeline captured by the stereo camera installed on the ego vehicle. Image on right is the extracted indicator signal shown by the green rounds.

where (x_J, y_J) is location of nearest junction with respect to the T_{car} extracted from the GIS map.

6. *Indicator signal I:* Prediction of a target car's behaviour using a known indicator signal will be more reliable and accurate [91]. If there is a junction nearby (extracted from road-network) and T_{car} car is giving a right indicator there is a high probability that the T_{car} will make a right turn. If the location is on a motorway and the left indicator is on there is high probability the car will merge to a left vacant lane. An example of indicator detection is shown in Figure. 2.26. Based on the extracted indicator signal, I is defined as

$$I = \begin{cases} -1, & \text{if Left Indicator is ON} \\ 1, & \text{if Right Indicator is ON} \\ 0, & \text{if both Indicators are OFF} \\ 2, & \text{if Hazard lights are On} \end{cases}$$

7. *Lane number or relative lane position L:* The position of the T_{car} with respect to ego-lane (where lane information is available) will help to predict future diversion. If T_{car} is within the lane, there is low probability that it will perform a lane change in a short future time horizon. In contrast if T_{car} is almost at the left side of the current lane or some portion of the car has already left the current lane and moved to the available left lane there is a high chances that it will merge to left lane in the immediate future. On the other hand, at a multi-lane junction, cars in the left-turn lane are more likely to make a left turn from the junction and vice-versa. On a motorway, lane location can play a crucial role in speed estimation. Extracted lane information is shown in Figure 2.27. Based on the extracted lane information L is defined as

Figure 2.27: Estimated lane T_{car} is currently in, is shown by red boundaries. Number at top of the bounding box is distance of the detected car from ego vehicle in metres. Two fractional numbers at bottom of the bounding box are the distance of T_{car} from the nearest lane markings respectively in metres. Dividing these two values will give the relative lane position.

$$L = \begin{cases} -2, & \text{if Left Turn \& Through Lane} \\ -1, & \text{if Left Turn Lane (Only)} \\ 0, & \text{if Through Lane} \\ 1, & \text{if Right Turn Lane (Only)} \\ 2, & \text{if Right Turn \& Through Lane} \end{cases}$$

2.7 Summary

Vehicle intention prediction for the safe navigation of autonomous vehicles is a well researched area. This chapter provides an in-depth survey of the existing prediction algorithms, categorized into three different categories, manoeuvre prediction, trajectory prediction and manoeuvre driven trajectory prediction. In the case of manoeuvre prediction the model is only responsible for prediction of the manoeuvre intention of the human driver, whereas trajectory prediction techniques produce the future trajectory. manoeuvre driven trajectory prediction is more attuned to human behaviour, as the model first predicts the future manoeuvre and then the future trajectory is predicted given past observations and the predicted maneuver. However, a human driver usually performs three steps while performing the ego-

vehicle path planning. First he/she decides the future maneuver, which is either left lane change, right lane change, straight on or right turn, left turn, straight on depending on the current context. Next he/she decides the motion or velocity at which the planned manoeuvre is executed and finally as result we see the future trajectory. In a machine-driven algorithm, the missing intermediate motion prediction stage in most of the existing methods makes it harder for the network to learn the end-to-end maneuver-driven vehicle kinematics. Existing techniques also tend to fix the surrounding vehicles at the beginning of the observation sequence. This makes it very hard to cope with a dynamic traffic scene where new vehicles may appear or existing vehicles may disappear throughout the entire past observation sequence as well as the prediction horizon.

During the early stages, various classical approaches such as Kalman filters and particle filters were used to perform the trajectory prediction task. Due to the missing surrounding vehicle interaction these models became very inaccurate during long term prediction. To address this problem various interactive models such as the Social Force Model [65], [127] and Intelligent Driver Model [96], [32] were used, but formulating the inter-vehicle force fields or developing smart driving models becomes very complicated with a growing number of surrounding vehicle in a dense traffic scene. Moreover, most of these models only consider the current state of the target vehicle and are unable to capture the inherent temporal features. This is a crucial factor for accurate long term trajectory prediction. This motivated a lot of researchers to formulate the trajectory prediction task as a sequence-to-sequence problem where, given a sequence of positions as input trajectory, the goal is to predict the future trajectory of all the vehicles in the scene as a future sequence of positions. As mentioned in Section 2.3 a lot of work has been done using an encoder-decoder architecture for sequence-to-sequence trajectory prediction problems. The main building block of any such encoder-decoder architecture is the RNN cell, as reviewed in Section 2.2.

The majority of the trajectory prediction models introduce context, such as the surrounding vehicle positions and map features as reviewed in section 2.6, only in the encoder and not in the decoder. The decoder is then only responsible for decoding the future position of the "target vehicle". The problem with this approach is that during long term prediction, the absence of any updated context or scene information makes the predicted trajectory inaccurate. This was the main motivation behind the development of a new LSTM network approach in which the surrounding vehicle positions and the context information are updated progressively in the decoder to keep the scene context relevant during long term prediction.

Even though the developed algorithms can be used for any sensor module, and indeed the developed model was tested on benchmark video sets collected from over-

head video cameras, this thesis also focuses on vehicle trajectories generated using an automotive radar due to its ability to work in adverse weather or lighting conditions. Hence, a specific section on detection, tracking, segmentation and behaviour prediction using automotive radars 2.4 was also included, mainly focused towards FMCW automotive radars. Keeping the other sensor modalities also in mind, section 2.6 explains how various surrounding vehicle features such as relative positions, relative speed, headway distance, indicator signal etc and map based context information such as distance from the nearest junction, lane based localization, lane rules (turn lane or through lane) can be extracted using a RGB-stereo camera and a GPS/IMU advance navigation tool kit.

Chapter 3

Interactive Vehicle Trajectory Prediction with Convolutional Recurrent Neural Networks

In this chapter, a novel conv-LSTM based trajectory prediction model is presented. The spatio-temporal features are captured simultaneously, and unlike previous techniques, the context information is made available not just in the encoder but also in the decoder to keep the scene information more up to date during long term prediction. The developed model has been evaluated against two publicly available benchmark datasets. Finally, section 3.7.3 explains how the lack of traffic codes in the prediction model leads to wrong trajectory prediction in two highly interactive scenarios.

3.1 Introduction

The two most crucial factors for future trajectory prediction are each vehicle's past movement i.e. temporal behaviour, and their relative position with respect to each other i.e. spatial geometry. In this context, the primary spatial geometry can also be thought of as inter-vehicle interactions. In most of the recent work these interactions were captured by first creating an OGM using the vehicles' relative positions. The same step was followed for a sequential OGM to generate the high level temporal features. These OGMs have been used in two ways to perform future trajectory prediction. In the first case, the entire OGM sequence was passed to a CNN based architecture which was trained to predict the future position of the target vehicle for the entire prediction horizon [117], [59], [71]. The issue with this approach is that the CNNs are developed mainly to extract spatial features and due to the missing interconnection between consecutive frames it became hard for the

model to understand the inherent temporal characteristics. In the second case the prediction was done using a two stage process [117], [59]. At first the OGM sequence was passed to a CNN architecture to extract the high level spatial features. The extracted features were then passed to an LSTM or GRU based recurrent neural network to finally predict the future position of the target vehicle. The problem with this approach is that the highly co-dependent spatio-temporal features were captured separately.

In our apprach both were captured simultaneously using a Conv-LSTM based deep learning model. Both the target vehicle (for which the prediction is being performed) and the surrounding vehicles in the vicinity were placed into a multi channel occupancy grid map (OGM) using their relative positions at the current frame. A sequence of OGMs was created to capture the temporal behaviour using all the vehicle positions in the previous frames. In contrast to the previous work, instead of passing individual OGMs into CNN based architectures and then to a recurrent neural network, the sequence of OGMs was fed directly into the developed and trained Conv-LSTM based architecture to extract the spatio-temporal features simultaneously and finally mapped to the target vehicle's position at the next frame.

Moreover, in most of the recent works the trajectory prediction problem has only been formulated as a sequence-input-sequence-output (SeqToSeq) structure. This means that the sequence of past positions for both the target and surrounding vehicles is fed to the network and the future trajectory i.e. sequence of positions only for the target vehicle is predicted. In this way the inter-vehicle interaction effect is only be present in the input sequence and not during the long term prediction sequence making the scene information outdated quickly. To handle this issue, the trajectory prediction problem was formulated as a sequence-input-single-output (SeqToOne) structure which means that instead of predicting the target vehicle's future movement for the entire prediction horizon it will predict the future position only at the next time instance, but for all the vehicles in the vicinity. Second, a novel feedback scheme has been developed that uses the currently predicted positions of all the vehicles to update the input OGM sequence before predicting for the next time instance. This keeps the scene information up to date, even during long-term prediction. Both the developed Conv-LSTM based architecture and the feedback scheme was bench-marked against the state-of-the-art networks using two publicly available datasets, i.e. NGSIM [24], [23] and High-D [89]. Moreover, in the case of NGSIM, the model was trained using US-101 highway data [23] and tested using I-80 highway data [24] to demonstrate the generality of the developed technique. In summary the major contributions of this chapter are as follows:

- A novel architecture was designed including Conv-LSTM [179] layers which

captures both the target and surrounding vehicle's spatial as well as temporal movement simultaneously for future movement prediction, since these two factors are highly co-dependent.

- To solve the vehicle association problem efficiently, separate Occupancy Grid Maps (OGM) are generated with respect to each target vehicle.
- To incorporate the social effect of surrounding vehicles' movement on a target vehicle's future motion, a novel feedback mechanism was also proposed to update the input OGMs after each prediction using the current predicted positions of all the surrounding cars.

3.2 System Model

The problem addressed in this chapter is as follows: given the past observation of the target vehicle as well as all the surrounding vehicles in the vicinity of the target vehicle the task is to predict the future movement of both the target and the surrounding vehicles. There are three key assumptions in this chapter. First, the observed vehicle trajectories must be perfect. This means there should not be any missed detections or occlusions causing incomplete trajectory data. Second, only those vehicles appearing at the start of the observation sequence along with the target vehicle are considered as surrounding vehicles. This is because the proposed interactive prediction technique in this chapter also predicts the surrounding vehicle's future movement and without a fixed length of past observation sequence it is impossible to do so. Third, due to the fixed shape and orientation of the created OGMs this technique will achieve best performance in straight roads specially multi-lane highways. The main motivation behind using a fixed shape and orientation OGM is to make it generally applicable to any type of highway with different numbers of lanes. Ideally, the trained model will be able to perform the trajectory prediction task in a completely new scenario, for which the model was never trained.

Moreover, a rectangular shape was adopted during the entire observation and prediction horizon as vehicles will travel a lot more in the forward direction compared to sideways movement. In order to preserve the relative movements between consecutive frames it is important to maintain a fixed origin for all the positions in both the observation and prediction time instances. With slight modification this technique can also be used at other traffic locations such as roundabouts or intersections. Then, the width of each OGM needs to be increased so that it can accommodate the starting point of all the side roads merging at a roundabout or at an intersection. Once the target vehicle completes the turn manoeuvre a new set

of OGM observation sequences will be created but this time the orientation will be aligned with the side road in the which the target vehicle entered.

Various map based features such as the number of lanes, presence of any intersection, type of the intersection i.e. signalled or unsignalled etc are also crucial factors for target vehicle's future trajectory prediction. Considering the approach presented in this chapter, such features must be encoded in individual OGMs, thus forcing the model to perform two additional tasks. First those features have to be extracted from the OGMs and then it is necessary to understand how they can impact the future movement of the target vehicle. The main focus of this chapter is to develop a model that can realize how the past relative movements of other vehicles in the vicinity can impact the target vehicle's future movement. This is why, in the work presented in this chapter, none of the map based features were used while creating the OGM observation sequence.

3.3 Problem Formulation

3.3.1 Vehicle Trajectory Representation

A vehicle trajectory is represented as a sequence of points, and each is associated with a unique time instance $[(x,y)_{t-h}, (x,y)_{t-h+1}, ..., (x,y)_t]$ where h is the length of the observation sequence and $(x,y)_t$ is the co-ordinate position at time t. For N vehicles in a scene, their trajectories T are represented as a 2D matrix below, where the ith row holds the trajectory of the ith vehicle along with the physical length (Len) and width (Wid) of the corresponding vehicle:

$$T = \begin{bmatrix} (x,y,Len,Wid)^1_{t-h} & (x,y,Len,Wid)^1_{t-h+1} & \dots & (x,y,Len,Wid)^1_t \\ (x,y,Len,Wid)^2_{t-h} & (x,y,Len,Wid)^2_{t-h+1} & \dots & (x,y,Len,Wid)^2_t \\ \dots & \dots & \dots & \dots \\ (x,y,Len,Wid)^N_{t-h} & (x,y,Len,Wid)^N_{t-h+1} & \dots & (x,y,Len,Wid)^N_t \end{bmatrix}$$

The goal is to predict the future trajectory $TPred$ of all vehicles for the next F time instances given T. $TPred$ can be given by

$$TPred = \begin{bmatrix} (x,y)^1_{t+1} & (x,y)^1_{t+2} & \dots & (x,y)^1_{t+F} \\ (x,y)^2_{t+1} & (x,y)^2_{t+2} & \dots & (x,y)^2_{t+F} \\ \dots & \dots & \dots & \dots \\ (x,y)^N_{t+1} & (x,y)^N_{t+2} & \dots & (x,y)^N_{t+F} \end{bmatrix}$$

3.3.2 Problem Statement

Hence, the goal is to predict the position of all the cars $(x^n_{t+1,t+F}, y^n_{t+1,t+F}) \forall\, n = 1, 2, 3.., N$ for the future F frames given the past positions of all the cars for the last

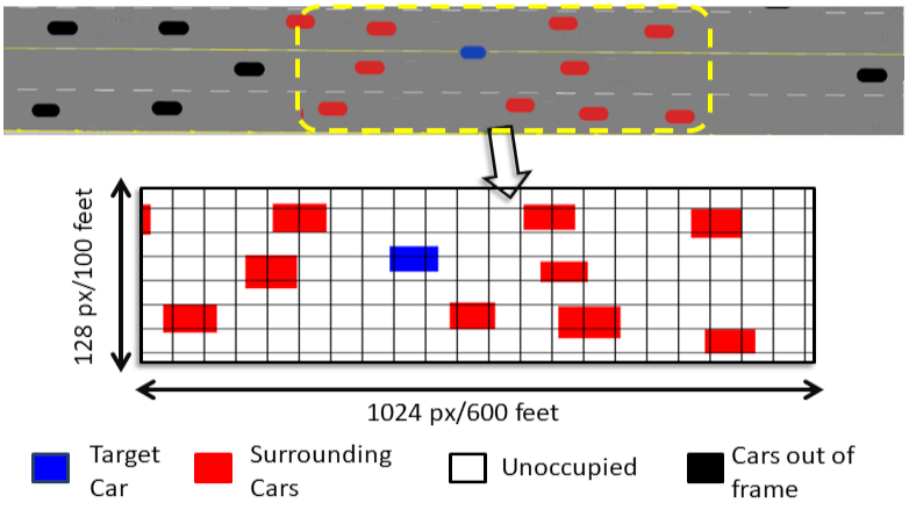


Figure 3.1: Generated occupancy grid map from the traffic scene with respect to a specific target car.

h frames. To achieve this the problem was further divided into two sub problems explained below,

- Predicting the next position $(x^n_{t+1}, y^n_{t+1}) \forall\, n = 1, 2, 3.., N$ of all the cars in the scene considering each as a target car for once, given the observation sequence $OGM^n_{t-h,t}$ created with respect to that specific car n.
- Updating each individual car's observation sequence $OGM^n_{t-h+f,t+f} \forall\, n = 1, 2, 3.., N\, and\, 1 \leq f \leq F$ given all the predicted positions (x^n_{t+f}, y^n_{t+f}).

3.3.3 Spatio-Temporal Scene Representation Using an OGM

Generally, within an Occupancy Grid Map (OGM), each cell value indicates the probability that cell is occupied. In this case, since the positions and trajectory information T are labelled manually a binary cell was considered which means that cells are either occupied ('1') or vacant ('0') as shown in Figure 3.1. Thus a single OGM will hold the spatial relationship between the target and surrounding cars and a sequence of these OGMs will hold the temporal information. The joint spatio-temporal features can be represented as a 4D matrix $OGM^{R\times C\times 2}_{t-h,t} \forall\, R, C, h > 0$ where R, C are the number of rows, columns of each OGM and h is the number of frames within the observation sequence and 2 is to maintain two separate channels of each OGM so that the target and surrounding vehicles can be placed separately. Separate observation sequences are created for each vehicle in the scene considering that particular vehicle as the target vehicle. $OGM^n_{t-h,t} \forall\, n = 1, 2, ..., N$ denotes a sample observation sequence created with position information from $t - h$ to t considering c as the target vehicle, where t is the current time instance. The origin of each observation sequence will be the position of the target vehicle at time t.

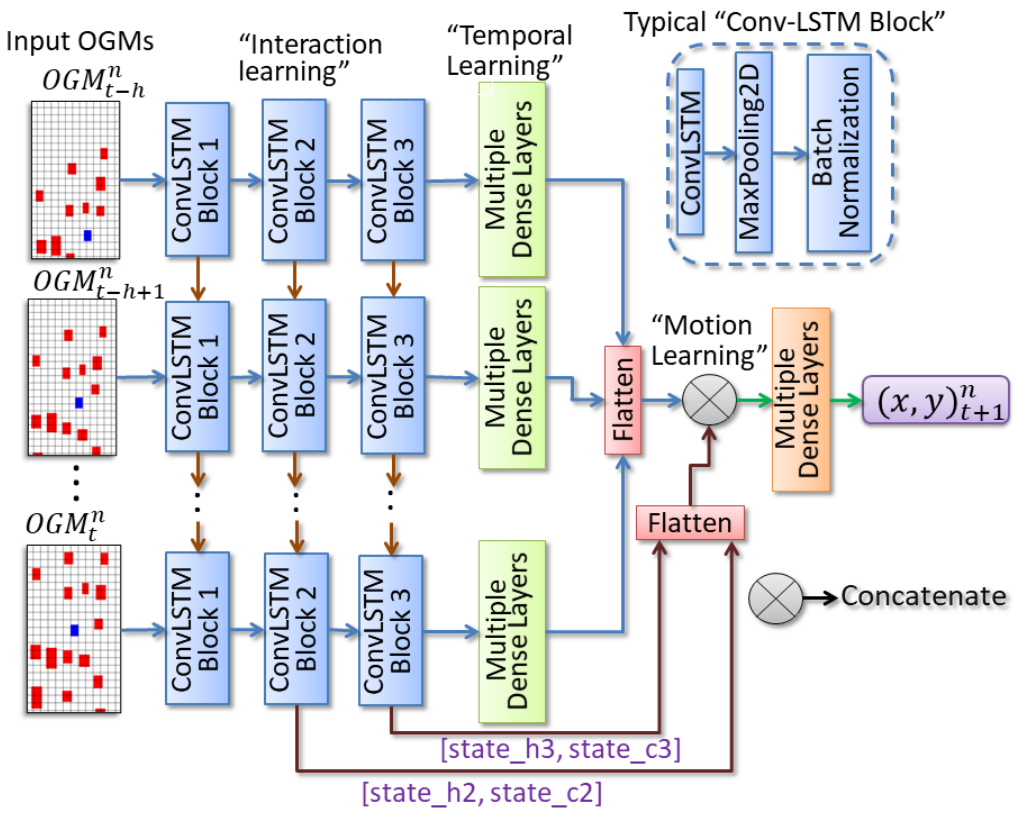


Figure 3.2: Proposed Conv-LSTM based architecture.

3.3.4 Frame of Reference

The origin of the all the grid maps in an observation sequence is the position of the target car at the beginning of the observation sequence. All the surrounding vehicles are placed on an individual OGM based on their relative positions with respect to the target car at that time instance. By maintaining a fixed origin of all the OGMs in the entire prediction as well as observation sequence, it is possible to keep both the target and the surrounding vehicles in a single coordinate frame making it easier for the model to capture how the past temporal features are correlated with future movement. On the other hand, with a moving OGM with respect to the target vehicle, the relative movement information would get lost. For example, consider if at the first time instance the target vehicle is at position (x_0, y_0) which is also the origin of the first OGM, and at the second time instance the target vehicle is at position (x_1, y_1) which is now the new origin of the second OGM. In both these OGMs, due to different origins, the target vehicle will appear at the same location making it difficult for the model to understand its relative movement from the last time instance to the current time instance.

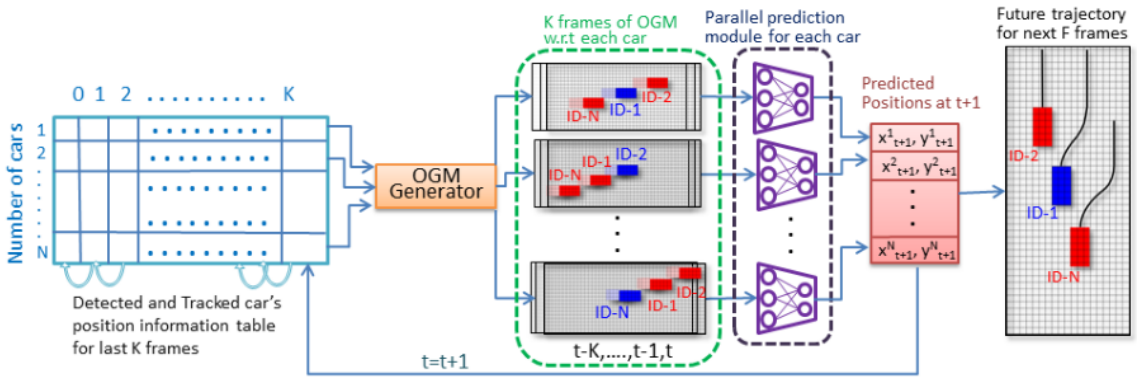


Figure 3.3: Proposed feedback prediction scheme.

3.4 Conv-LSTM Based Future Trajectory Prediction

Recurrent neural networks are designed for sequential problems, specifically when dealing with temporal information. However vanilla RNN [73] and LSTM [57], [28] networks are incapable of using both the temporal and spatial information simultaneously. To address this problem a Conv-LSTM deep network architecture [179] has been used in this work.

3.4.1 Proposed Network Architecture

The proposed network architecture consists of three modules, Interaction Learning, Temporal Learning and Motion Learning. It is shown in Figure 3.2. $OGM^{n}_{(t,t-h)}$ are the input OGMs at time t for vehicle n. The blue "Conv-LSTM" blocks are part of the "Interaction Learning" module, green "Multiple Dense Layers" are part of the "Temporal learning" module, finally the grey "Concatenate" layer is the "Motion Learning" module which concatenates the states from the the "Conv-LSTM" blocks and also outputs from the "Multiple Dense Layers" and passes it to another set of "Multiple Dense" layers to finally predict the position of vehicle n at the next frame, $(x, y)^{n}_{t+1}$.

Interaction Learning

The Interaction Learning module learns the dependency between the movement of target vehicle, c, and its surrounding vehicles. It takes $\mathbf{OGM}^{c}_{t-h,t}$ as input. As shown in Figure 3.2, this part has three "ConvLSTM Blocks", each of which is composed of a 2D Conv-LSTM layer followed by MaxPooling and BatchNormalization layers. The kernel dimensions in the three different Conv-LSTM layers are (3,3), (5,5) and (7,7). The small dimension in the first layer is to successfully extract the relative car positions. In the following two layers have different kernel dimensions to separate

Table 3.1: Detailed architecture of the proposed Interaction-Learning Module including layer type, filter count, activation functions and kernel dimensions.

Layer Type	Activation	Alpha	Kernel	Filters
Conv-LSTM	Leaky-Relu	0.1	(3,3)	2
BatchNormalization	---	---	---	---
MaxPooling	---	---	(3,3)	---
Conv-LSTM	Leaky-Relu	0.1	(5,5)	4
BatchNormalization	---	---	---	---
MaxPooling	---	---	(5,5)	---
Conv-LSTM	Leaky-Relu	0.1	(7,7)	8
BatchNormalization	---	---	---	---
MaxPooling	---	---	(7,7)	---

the dependency of near and distant cars on the target car's future movement. To maintain the OGM dimension, the "same" padding is employed to all three Conv-LSTM layers. The filter counts for the three Conv-LSTM layers from the beginning are 2, 4 and 8, respectively, with Leaky-ReLU (Leaky-Rectified Linear Unit) as the activation function to avoid neuron death. A detailed structure of the interaction-learning module is mentioned in Table 3.1.

Temporal Learning

Recent frames within the historical observation sequence are more informative than older frames for future motion estimation. To capture this varied contribution of individual frames separate sets of fully connected layers are maintained for each temporal instance. Extracted features from each temporal Conv-LSTM slice are then fed directly to the fully connected layers. The sizes of the layers in this module are $128 \rightarrow 64 \rightarrow 32 \rightarrow 16$ with Leaky-ReLU as the activation function.

Motion Learning

The last part of the model is responsible for motion learning based on the extracted features from the previous two parts. All the temporal slices were concatenated with the extracted state from the Conv-LSTM layers and fed through fully connected layers to learn the correlation between previously extracted features and future movement in the next frame. The neuron counts of the layers are $128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 2$ with Leaky-ReLU as the activation function except the last layer. As the underlying task is regression linear activation was used in the last layer to predict the 2D position of the target car in the next frame.

Table 3.2: Detailed architecture of the proposed Temporal and Motion Learning Modules including layer type, neuron count, activation function and alpha.

Module	Layer Type	Neurons	Activation	Alpha
Temporal-Learning	Dense	128	Leaky-ReLU	0.1
	Dense	64	Leaky-ReLU	0.1
	Dense	32	Leaky-ReLU	0.1
	Dense	16	Leaky-ReLU	0.1
Motion Learning	Dense	128	Leaky-ReLU	0.1
	Dense	64	Leaky-ReLU	0.1
	Dense	32	Leaky-ReLU	0.1
	Dense	16	Leaky-ReLU	0.1
	Dense	8	Leaky-ReLU	0.1
	Dense	2	linear	---

A detailed description of both the temporal and motion learning modules are mentioned in Table 3.2.

3.4.2 Feedback Based Prediction Technique

As mentioned earlier, the model is trained with the last h OGM frames as input and the position of target car in the next frame as output. To update the positions of the target and surrounding vehicles in the observation sequence after each instance prediction, a feedback based scheme shown in Figure 3.3 is introduced. The vehicle position table (left, blue) holds the historical position in the last h frames for each car in the scene. The orange OGM Generator block is responsible for N observation sequence generation considering each vehicle as the target vehicle once. The N violet Prediction blocks predict the position $(x, y)_{t+1}$ in the next frame. Finally, the predicted positions are stored (the last red box) and fed back to the position information table (first blue table) to update the current scene simultaneously.

The first part is the vehicle past position table of shape (N,h) which holds the positions of each car for the previous frames from $t - h$ to t from trajectories T. Whenever there is a "birth" or the appearance of a new car in the scene, it appends its location at the end of the table. Similarly whenever there is a "death" or a car goes out of the scene, the row associated with that car is removed. This vehicle position table is then passed to the OGM Generator block which creates N observation sequences considering each vehicle as a target vehicle once. Each observation sequence is then passed through the pre-trained model to predict the next position $(x, y)_{t+1}$ at $t+1$. Once all the vehicle's future positions are predicted,

they are concatenated to represent the full set prediction at $t+1$. This updates the vehicle position table by replacing the oldest frame. After the first prediction the vehicle position table holds position information from $t-h+1$ to $t+1$ for each vehicle. Once new positions are appended to the vehicle position table, their OGMs are generated through the OGM generator block. Similarly, after the second prediction, the position table has the observation from $t-h+2$ to $t+2$. The process is repeated until it reaches the maximum prediction horizon F.

The steps involved in predicting the positions of each vehicle at each future timestamp for all the vehicles currently present in the scene and recreating the current scene information using those predicted positions is summarised in Algorithm 1 During future movement prediction of one particular vehicle at $t+f$, $\forall f = 2, 3, ..., F$ where f is the current prediction instance, the predicted positions of all the other cars between $t+1$ and $t+f-1$ are also considered in the observation sequence. Hence, this feedback updates the input sequence repeatedly to incorporate potential future interactions with other cars more effectively.

3.5 Implementation Details

This section explains data formatting, training, and validation on two publicly available datasets i.e. NGSIM [23], [24] and High-D [89]. Details about both these datasets are presented in Sections 3.6.1 and 3.6.2.

3.5.1 Data Formatting

The pixel dimensions of the OGM considered in this work are 1024 rows in the longitudinal and 128 columns in the lateral direction with 3 ($h = 30$ frames) and 5 ($F = 50$ frames) seconds as the observation and prediction time windows, respectively. The physical dimensions of the grid map are 600 feet (∼180 m) by 100 feet (∼30 m). In both NGSIM (US101 [23] and I80 [24]) and High-D [89] datasets, among all the tracked vehicles the highest velocity recorded is roughly 75 feet/sec and the total observation and prediction temporal windows in this experiment are 3 + 5 = 8 secs which allows each car to move roughly 600 feet, which sets the longitudinal physical dimension. In the lateral direction, a 100 feet wide grid map covered the adjacent lanes along with the target car's ego-lane with some extra margin on both sides. So one single OGM can hold the spatial relationships between all the cars and since both NGSIM and High-D have a data acquisition rate of 10Hz, 30 sequential OGMs will jointly represent the spatio-temporal information for the 3 seconds observation window. In the longitudinal direction 600 feet physical length were quantized using 1024 rows which led to approximately 600/1024 = 0.58 ft/pix

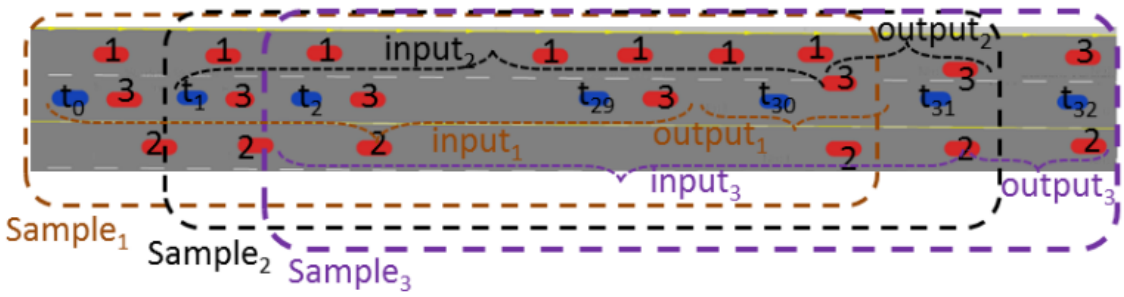


Figure 3.4: Sliding window sampling scheme for training the network. The blue and the red are the target and surrounding vehicles respectively. Multiple copies of the same vehicle id indicates different time instances.

OGM resolution. On the other hand in case of the lateral direction 128 feet physical width which includes roughly 2 adjacent lanes on both sides along with the target vehicle's ego lane, were quantized using 128 columns leading to 100/128 = 0.78 ft/pix OGM resolution. In any highway scenario vehicles tend to accelerate or decelerate more often compare to performing a lane change maneuver. Considering the orientation of the created OGMs any vehicle acceleration or deceleration will lead to position change in the longitudinal direction and any lane change manoeuvre will lead to position change in the lateral direction. This is why during the OGM creation longitudinal resolution was maintained higher compare to the lateral resolution. So that the memory consumption for each input OGM can be reduced without loosing the raw trajectory information much. Moreover, while deciding these dimensions the numbers that are power of 2 were only considered to optimize the use of memory. Keeping this in mind along with the physical length and width of actual vehicles any higher number of rows or columns for the OGMs will become redundant in terms of trajectory accuracy. For each OGM generation the locations estimated through the global state plane co-ordinate system were used instead of the US-101 [23] or I-80 [24] specific local co-ordinate system to make it more generic. The future trajectories were predicted with respect to the local coordinate system. Once predicted, it can be converted back to the global co-ordinate system with the knowledge of its origin location in the state plane co-ordinate system.

3.5.2 Multi-Channel OGM

Instead of placing both the target and surrounding vehicles into the same OGM double-channel OGMs were created to pass them separately to the network. That made it easier for the network to differentiate the target vehicle for which the prediction is being performed and the surrounding vehicles whose relative positions are affecting the target vehicle's future motion. The first channel of all the OGMs in

one observation sequence holds just the relative movement of the target car from start to end of the observation sequence and the second channel holds the relative movement of all the surrounding cars with respect to the target car for the same sequence. Vehicle dimensions are usually 8 ft/16 ft for cars and 14ft/54ft for HGVs. Considering these dimensions and the feet per pixel resolution of the OGMs each vehicle will roughly have 250-1500 pixels. The input data can be think of as a two-channel birds-eye view image of the current scene.

For faster and efficient sample generation at first two dictionaries were created, "Vehicle_Dict" and "Frame_Dict" from the entire NGSIM and High-D dataset with Vehicle_ID and Frame_ID as the keys, respectively. Each vehicle in the Vehicle_Dict was selected as target car once and the current Frame_ID of the selected target vehicle along with the Frame_Dict was jointly used to identify the surrounding vehicles present at that instance to create the input trajectory matrix T mentioned in the vehicle trajectory representation. Due to the fixed input shape of the proposed network only the vehicles present in the last $h = 30$ frames were considered to create T. Once created each vehicle in the trajectory matrix (T) was considered as a target vehicle once to create its OGMs for the entire observation sequence ($h = 30$ OGMs). The detailed steps involved for the OGM creation for each target car are given in Algorithm 2. A sample multi-channel OGM is shown in Figure 3.1.

3.5.3 Sampling the Sequence

A sliding window is used to generate sequence samples which can be used for the SeqToOne architecture. Consider generating samples for one specific target vehicle (the blue vehicle in Figure 3.4) with surrounding vehicles 1,2 and 3 (the red cars in Figure 3.4). To generate the first sample ($Sample_1$) of the observation sequence associated with the target vehicle, it uses its historical position information from t_0 to t_{29} (for $h = 30$ frames) with the corresponding surrounding cars in the same co-ordinate frame considering the target vehicle's position at t_{29} as origin. The target car's position at t_{30} in the same co-ordinate frame will be the ground truth output. For the next sample ($Sample_2$), for the same target vehicle the time window was moved one unit forward, which means the observation sequence will be generated based on the target vehicle's position from t_1 to t_{30} considering the position at t_{30} as the origin and the position at t_{31} as the output. Using the same time window shift approach, the rest of the samples can be generated.

This kind of sampling scheme helps the feedback based technique to predict accurately by providing similar data points during training. Now let's assume the trained model is predicting a different target car's future movement in a similar traffic situation using the feedback technique. Input would be the observation se-

quence of 30 OGMs generated using the target car's position from t_0 to t_{29} and the predicted output would be the position at t_{30} which is similar to $sample_1$ in the training dataset. Once predicted for t_{30}, the predicted position along with positions from t_1 to t_{29} is used to predict the position in the subsequent frame (t_{31}). Due to the sliding window sampling technique, this time it will find correspondence with $sample_2$ and will predict accurately. In the same way for the next frame (t_{32}) it will find a strong similarity with $sample_3$ and so on.

3.5.4 Training

The Conv-LSTM models are trained by minimizing the Euclidean distance between the predicted and the ground truth positions over all the samples:

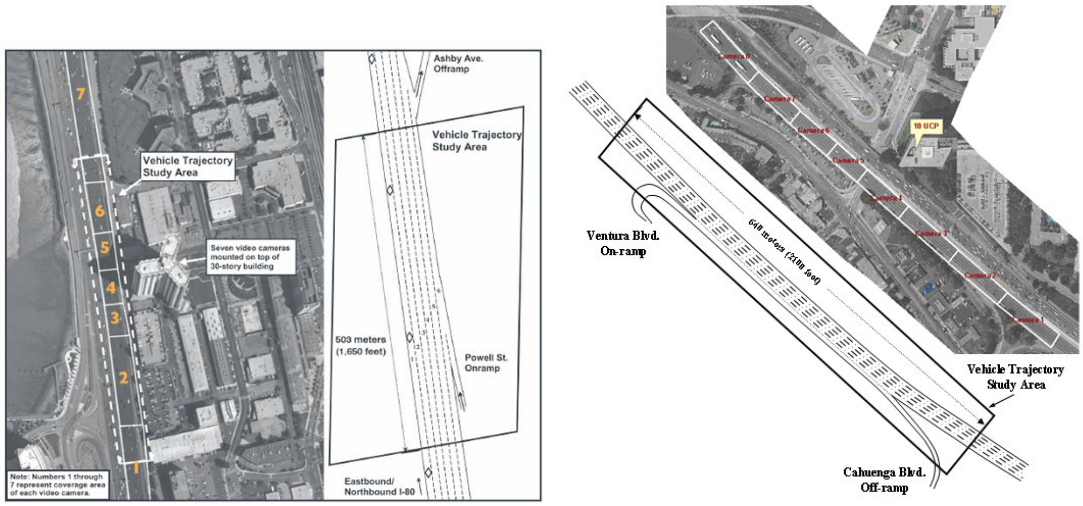
$$\mathcal{L} = \sqrt{(x_{true} - x_{pred})^2 + (y_{true} - y_{pred})^2} \tag{3.1}$$

where $\mathcal{L}$ is the computed loss, x_{true} and y_{true} are the ground truth, and x_{pred} and y_{pred} are the predicted lateral and longitudinal positions, respectively.

The developed model was trained using the RMSprop optimizer [151] as this works better for recurrent models. The step decay learning rate starts from 0.01 and reduces by 30 percent after each 5 epochs. The model is implemented in Keras [85] with the TensorFlow backend [1], [186]. The implementation of the proposed technique can be found here.

3.6 Benchmark Datasets

One of the most popular and the first bird's-eye view dataset is the Next-Generation Simulation (NGSIM) dataset that includes 45 minutes of vehicle tracking data for two multi-lane intersections, the Lankershim Boulevard [25] and Peachtree Street [26], and two multi-lane motorways, I-80 [24] and US-101 [23], in the United States. Birds eye views of both the multi-lane highways and multi-lane intersections are shown in Figure 3.5 and Figure 4.6, respectively. The coverage area, collection sites, number of intersections and lane counts for all four sub-datasets in the NGSIM dataset are given in Table 3.3. Each sub-dataset includes more than 2400 trajectories of real traffic captured by video cameras mounted on the roof of a 36-storey building, with camera frame rates at 10 Hz for 45 minutes in three different traffic conditions, i.e. mild, moderate and congested. Each data point in the trajectory information includes lateral and longitudinal position, instantaneous speed, acceleration, current section ID, current intersection ID and instantaneous movement. The lateral and longitudinal positions are actually measured in the local co-ordinate system. In Figure 3.5(a) and 3.5(b) the origin locations are the bottom-left corner



(a) I-80 Highway [24]

(b) US-101 Highway [23]

Figure 3.5: Bird's Eye View (BEV) of two highway locations in the NGSIM dataset, i.e. I-80 and US-101. Each sub figure consists of a satellite view of the observation area and a schematic diagram with lane counts

of the black boxes drawn on top of the individual schematic diagrams. Instantaneous movement indicates which maneuver (straight-on or left/right turn) a vehicle is currently performing. A slightly different dataset was collected and published by Krajewski *et. al.* called the High-D dataset [89]. In this case instead of using fixed overhead cameras, an unmanned aerial vehicle (UAV) was used to collect vehicle trajectory data from 6 different highways in Germany with different lane numbers and speed limits. It includes real vehicle trajectories of more than 110500 vehicles where state-of-the-art computer vision algorithms were used to detect and localize each vehicle. The generated trajectories were further processed using Bayesian smoothing. The information associated with each trajectory is similar to the NGSIM dataset. Further details about both the NGSIM and High-D datasets are explained in Section 3.6.1 and 3.6.2, respectively.

3.6.1 Next Generation Simulation (NGSIM)

The next Generation Simulation (NGSIM) US-101 [23] and I-80 [24] highway datasets were collected and published by the US Federal Highway Administration. US-101 covers a 640 meters (2,100 feet) long, 5 lane wide (3.66m or 12ft each) section on a Hollywood freeway in Los Angeles (see Figure 3.5(b)) and I-80 covers a 503 meters (1,650 feet) long, 6 lane wide (3.66m or 12ft each) section on the I-80 freeway in Emeryville, California (see Figure 3.5(a)). These separate scenarios allow us to train and test the model on two completely different datasets. Each consists of more than 5000 trajectories of real traffic captured at 10Hz for 45 minutes with three different

Table 3.3: Details about four different sites in NGSIM dataset

Location	Coverage (m/feet)	Lanes	Intersections
I-80, Hollywood Freeway, Los Angeles	503/1650	5	–
US-101 Emeryville, California	640/2100	6	–
Lankershim Boulevard, Los Angeles	500/1600	3-4	4
Peachtree Street Atlanta	500/2100	2-3	5

traffic conditions, i.e. mild, moderate and congested. Each data point in the US-101 [23] and I-80 [24] datasets have Vehicle_ID (unique ID of the current tracked vehicle), Frame_ID (the current frame number after the vehicle has appeared in the installed video camera's field-of-view), Total_Frames (total number of frames the vehicle was present in the video camera's field-of-view), Global_Time (current Unix time), Local_X and Local_Y (position of the vehicle in the local collection area), Global_X and Global_Y (position of the vehicle in the global State Plane Co-ordinate System), V_Length and V_Width (length and width of the vehicle in feet), V_Class (type of the vehicle i.e. truck, bus, or car), V_Vel and V_Acc (instantaneous speed and acceleration of the vehicle respectively) and Lane_ID (ID of the current lane the vehicle is travelling in).

3.6.2 High-D Dataset

High-D [89] is a real vehicle trajectory dataset including more than 110,500 vehicles collected on 6 different highways with different lane numbers and speed limits using an unmanned aerial vehicle, shown in Figure 3.6. State-of-the-art computer vision algorithms were used to detect and localize each vehicle. The generated trajectories were further processed using Bayesian smoothing. The information associated with each trajectory is similar to the NGSIM dataset with some additional traffic features, i.e. lateral (x), longitudinal (y) positions (see Figure 3.6 for x/lateral and y/longitudinal directions), instantaneous speed and acceleration (both in x and y direction), vehicle length and width (in meters), vehicle type (i.e. truck, bus or car), Lane ID (ID of the current lane the vehicle is travelling in), Distance Headway (DHW), Time Headway (THW) and Time to Collision (TTC) of preceding and following vehicles on the ego and adjacent lanes along with their IDs.

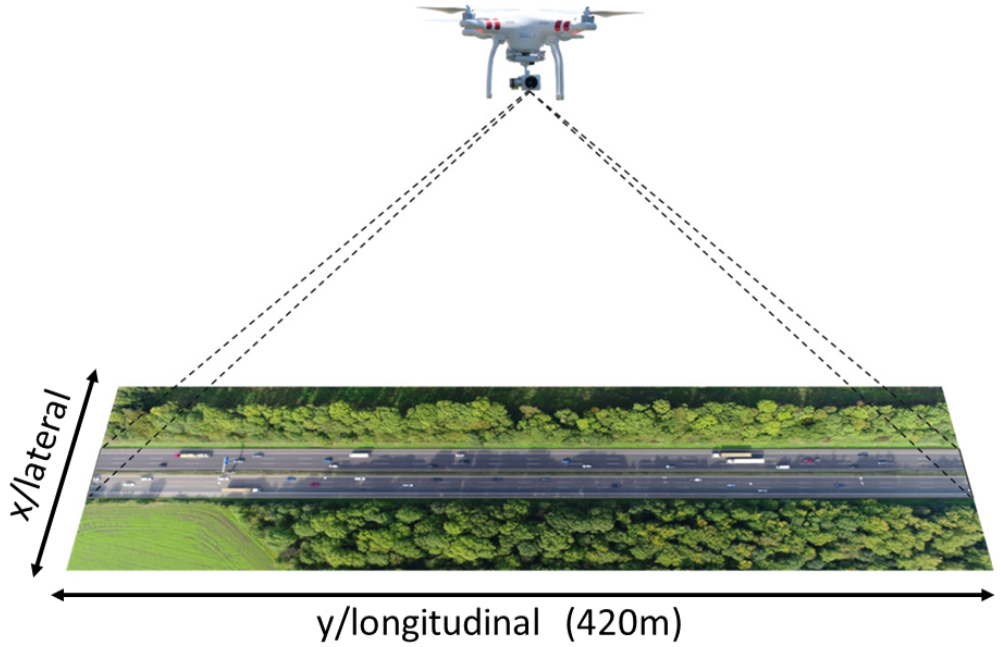


Figure 3.6: A example scene of the High-D dataset [89] collected with a drone (unmanned aerial vehicle) that captures traffic from a bird's eye view on a road section with a length of about 420 m.

3.7 Results and Performance Evaluation

3.7.1 Compared Models

The developed model was then compared with several existing models (defined below) either re-coding (CV,V-LSTM), downloading public code (CS-LSTM), or using published results (D-LSTM).

- **Constant Velocity (CV):** In the constant velocity (CV) model, only the instantaneous velocity of the target car was used to calculate the succeeding longitudinal and lateral positions.

- **Vanilla-LSTM (V-LSTM):** In the V-LSTM model only the target car's past trajectory was used to predict the future trajectory, not using any information about the surrounding cars. To ensure fair comparison, this model was also trained in SeqToOne fashion and then used the proposed feedback scheme to predict the whole future sequence. The reason the developed model was compared with V-LSTM is to understand whether the proposed technique can capture and benefit from other surrounding cars' information to make the long term prediction more accurate.

- **Dual-LSTM (D-LSTM):** A two-stage LSTM based network is proposed in [176] by Long *et al.* The proposed method feeds the input trajectory sequence

Table 3.4: RMSE comparison on NGSIM dataset [23] between CV, V-LSTM, Dual-LSTM, CS-LSTM and the proposed technique at different time horizons. The Dual-LSTM results are from the original paper [176] as the code or model is not publicly available. All the RMSE values are in metres.

	Prediction Horizon									
Model	**1 s**	**2 s**	**3 s**	**4 s**	**5 s**	**6 s**	**7 s**	**8 s**	**9 s**	**10 s**
CV	0.74	2.05	3.72	5.65	7.62	9.64	11.75	13.98	16.28	18.47
V-LSTM	0.81	1.78	2.85	4.06	5.39	6.84	8.38	10.05	11.85	13.77
Dual-LSTM [176]	0.47	1.39	2.57	4.04	5.77	–	–	–	–	–
CS-LSTM [29]	0.58	1.26	2.11	3.18	4.53	6.21	8.42	11.12	14.20	17.90
Proposed	0.80	1.67	2.39	3.23	4.50	5.66	6.92	8.43	9.73	11.40

to the first LSTM block to recognize the driver intention as an intermediate step. The second LSTM block receives the recognized driver intention to estimate the future trajectory. Since the code is not available and the model is tested on the same dataset the results provided in the paper were considered directly.

- **Convolutional Social Pooling-LSTM (CS-LSTM):** This incorporates a manoeuvre based (Left Lane Change, Right Lane Change or Follow Road) motion model to generate a multi-modal predictive distribution [29] using the conventional SeqToSeq encoder-decoder architecture. Even though it includes the surrounding vehicle interaction effect in the encoder but the decoder is only responsible to decode the future positions of the target vehicle only, making the scene information outdated during the long term prediction.

3.7.2 Performance Comparison with State-of-the-Art Methods

A comparison of performance of all algorithms over 4000 test sequences randomly selected from the NGSIM dataset [23] is shown in Figure 3.7. As a performance metric, the Root Mean Squared Error (RMSE) between the predicted and ground-truth positions were computed for a future horizon of 1 to 10 seconds for all the traffic participants (cars and HGV). In most of the recent literature the considered prediction horizon is only 5 secs but in this chapter it was extended to 10 secs

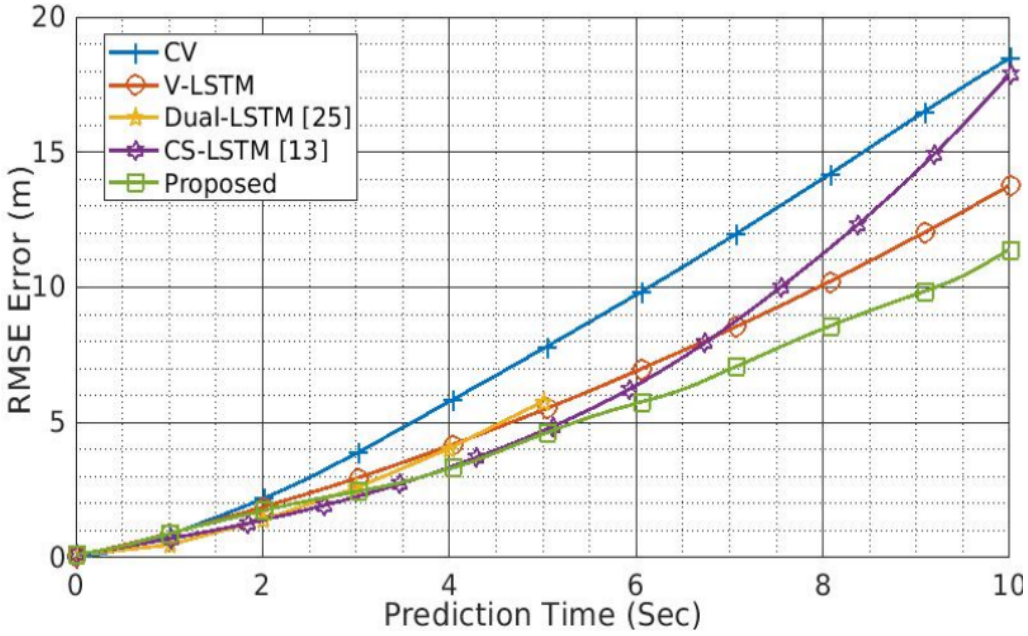
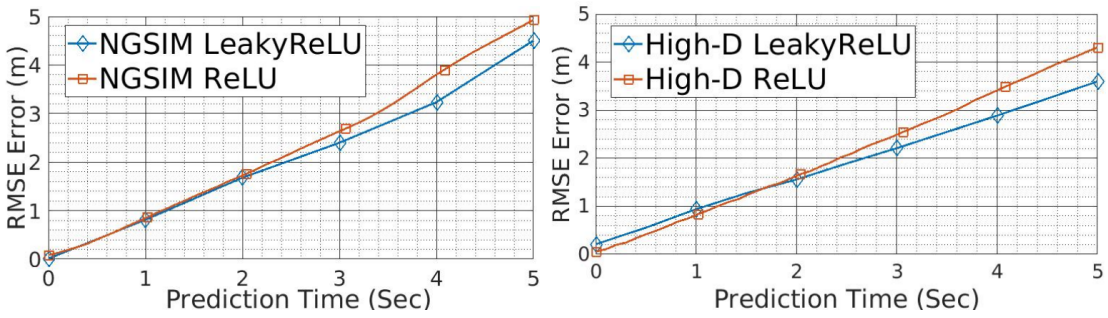


Figure 3.7: The RMSE comparison of CV, V-LSTM, Dual-LSTM [176], CS-LSTM [29] and the proposed method on 4000 sequences selected from NGSIM US-101 [23] dataset. This figure shows their average mean squared errors for the prediction time horizon from 1s to 10s.

to better understand how the proposed interactive prediction mechanism technique can keep the scene information up to date during a longer term prediction. While predicting the immediate future the observed surrounding scene information plays a crucial role, but while predicting in the longer term the observed sequence becomes outdated. Therefore, the predicted positions of all the surrounding vehicles as well as the target vehicle should be used as the current context. Although this model was only evaluated using the publicly available multi-lane highway datasets to compare its performance with existing algorithms, the main use of this type of technique with a long term prediction capability should be in a single carriageway scenario, especially when a vehicle is performing an overtake maneuver. The overtaking vehicle has to go onto the other side of the road where the vehicles are travelling in the opposite direction which means the relative speed is a lot higher and more dangerous. Before starting to overtake a human driver always anticipates the entire manoeuvre, crossing to the other side of the road, then merging back in. Depending on the velocity and length of both the overtaking vehicle and the vehicle in front, it can take typically more than 5 secs, especially if both are HGVs and the velocity difference between them is low.

It can be seen that the naive CV model produces the highest prediction error due to the absence of any temporal or interaction information. In V-LSTM, the past trajectory information was also employed, which makes future motion estimation (velocity and acceleration) more accurate and so outperforms the basic CV model.



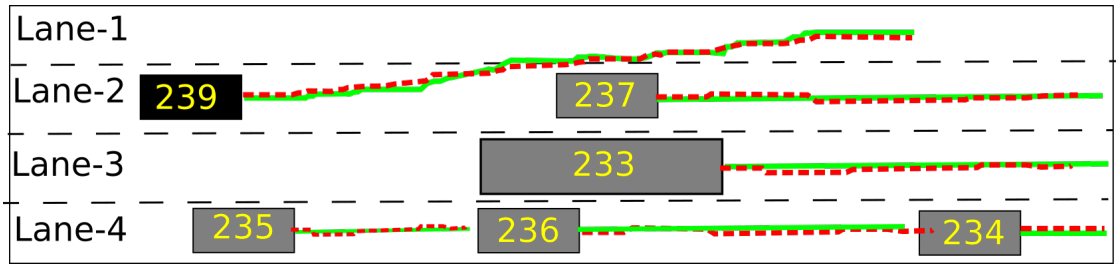
(a) NGSIM [23] (b) High-D [89]

Figure 3.8: The RMSE on NGSIM [23], [24] and High-D [89] datasets with two different activation functions (ReLU and LeakyReLU) for Conv-LSTM layers. For each evaluation metric, its average values were plotted for the prediction time horizon from 1s to 5s.

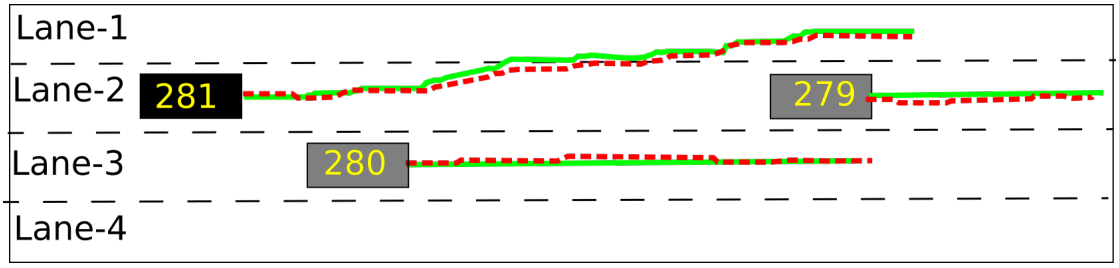
The developed model in this chapter includes the information on surrounding cars for the whole future prediction horizon outperforms both CV and V-LSTM; this suggests that the surrounding cars' relative motion plays a crucial role in future trajectory prediction.

The performance of Dual-LSTM is better than both the CV and V-LSTM models during the short term horizon due to the presence of the intermediate intention recognition step, but it is still not as good as CS-LSTM or the proposed method due to the missing surrounding cars' information. CS-LSTM (with manoeuvre information) and the proposed technique (without manoeuvre information) perform almost similarly during the short term future horizon. Achieving similar performance without manoeuvre class information (Lane Change, Follow Road) does save the effort and necessity of labeling each trajectory sequence during the training process. In addition, since this method also considers the predicted positions of all other surrounding vehicles in the scene to update the OGM frames, it outperformed the state-of-the-art CS-LSTM method during the long term prediction horizon. The RMSE of each method at different time horizons is shown in Table 3.4.

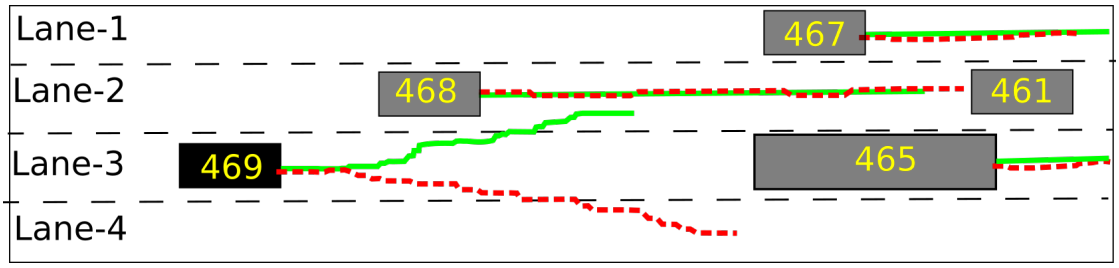
Two different activation functions, ReLU and Leaky-ReLU, were used for the first three Conv-LSTM layers to identify the contribution of activation functions to model performance. The comparisons of these two settings on two different datasets, NGSIM and High-D, are shown in Figure 3.8(a) and Figure 3.8(b), respectively. The performance with the Leaky-ReLU activation function is better than normal ReLU on both datasets. Moreover, due to the better variety and more natural trajectories, the overall performance on the High-D data is better than the NGSIM dataset. The RMSEs of each setting on each dataset, at different time horizons, are shown in Table 3.5.



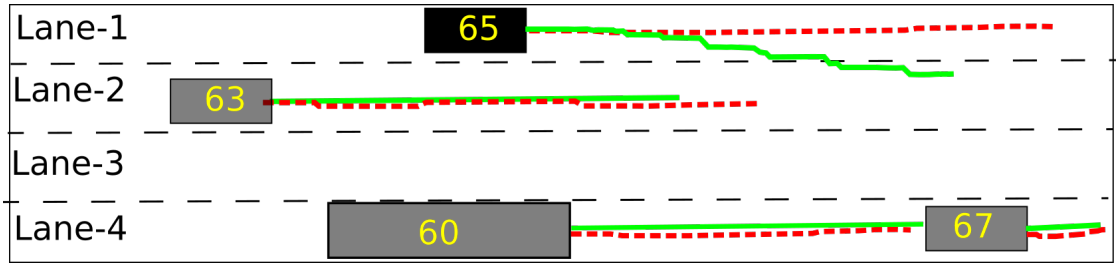
(a) Successful left lane change prediction on NGSIM dataset



(b) Successful left lane change prediction on High-D dataset



(c) Wrongly anticipated right lane change on High-D dataset



(d) Missed pulling back to right lane after overtake on High-D dataset

Figure 3.9: Case studies of the prediction results by the proposed method. The predicted and ground truth trajectories are drawn in dashed Red and solid Green lines respectively. Black dashed lines are the lane markings where Lane 1 is the left most lane. Each image shows future trajectory up to 5s.

Table 3.5: RMSE comparison on two different datasets (NGSIM and High-D) with two different activation functions (ReLU and Leaky-ReLU) for the Conv-LSTM layers at 5 different time horizons

		Prediction Horizon				
Dataset	**Activation**	**1 s**	**2 s**	**3 s**	**4 s**	**5 s**
NGSIM	ReLU	0.76	1.66	2.59	3.77	4.92
NGSIM	Leaky-ReLU	0.87	1.76	2.58	3.34	4.14
High-D	ReLU	0.74	1.57	2.44	3.39	4.30
High-D	Leaky ReLU	0.87	1.55	2.20	2.88	3.59

3.7.3 Case Studies

The distribution of different maneuvers in most recent datasets is heavily biased towards follow road sequences or in other words an absence of frequent lane changes. To address this problem the results were further illustrated by showing the predicted trajectories (see Figure 4.18) in four highly interactive scenes with a potential lane change. Figure 3.9(a) shows that target vehicle 239 is preceded by a relatively slow-moving vehicle 237 within the same lane and the right adjacent lane is occupied by HGV 233. In this scenario, ideal driving behaviour would be to perform a left lane change to overtake the slow preceding vehicle provided a sufficient gap is available in the left lane which is indeed the case. Another similar scenario in the High-D dataset with a high chance of lane change is shown in Figure 3.9(b) where vehicle 281 is preceded by slow-moving vehicle 279, the right lane is occupied by another vehicle 280, and the left lane is available. In both these cases the model successfully predicted a left lane change trajectory.

Within the US driving code, the overtake manoeuvre should only be performed using the left vacant lane; in the case when the left lane is occupied the driver should wait until it is clear. In addition, once the overtake manoeuvre is completed the vehicle should cut back in to the rightmost vacant lane. Contradictory trajectories, overtaking using the right lane and not pulling back into the right lane after successful overtaking, are shown in Figure 3.9(c) and Figure 3.9(d) respectively. Figure 3.9(c) shows the violation of the first code, where vehicle 469's ego lane and both the left lanes are congested. Due to the availability of a significant gap in the rightmost lane, the proposed model predicted a dangerous accelerated right lane trajectory as opposed to the decelerated left lane change ground truth trajectory. Figure 3.9(d) shows a violation of the second code, where vehicle 65's ego lane is empty and there is no need for an immediate lane change in terms of lane congestion. This is why the

proposed technique predicted a follow lane trajectory as opposed to the right lane change (ground truth) trajectory, pulling back into the rightmost vacant lane. There is a strong possibility that these violations are caused by the missing traffic code information in the model. Future work should consider adding rule-based layers to the model to handle these types of case.

3.8 Summary

In this chapter, a novel Conv-LSTM based architecture was proposed with an interactive feedback based prediction scheme to forecast the future trajectory of traffic participants from the current scene. The model has the capability to make use of both the spatial and temporal features simultaneously and thus improves the accuracy of the predicted trajectory. Moreover, the designed feedback scheme encodes the surrounding vehicle interaction, not only in the input sequence but also during the prediction horizon, by sequentially predicting future positions of all the surrounding vehicles in the vicinity and then using those predicted positions while predicting the next time instance. Several experiments on two different datasets, NGSIM [24], [23] and High-D [89], show that the proposed method can achieve comparable prediction accuracy with the state-of-the-art models during the long term prediction horizon, even without using manoeuvre information. Moreover, for both the datasets the proposed network was trained and tested on different highways to establish the generality of it. In the proposed feedback scheme, since the predicted position of the current frame was used to predict the position in the next frame, there is a chance of error accumulation during the long-term prediction horizon. This is why in the the next chapter a novel "retrain technique" has been used to reduce this error accumulation even further. In addition to this, various map based features were also introduced in the observation sequence to handle the wrongly anticipated trajectories that occur due to the missing contextual information.

Chapter 4

Predicting Vehicle Behaviour using Automotive Radar and Recurrent Neural Networks

In contrast to the previous chapter, the scene information, together with various map based features, are encoded in a feature vector formation instead of the high dimensional occupancy grid maps. Unlike existing models, the context information of the surrounding vehicles is not fixed at the beginning of the observation sequence but dynamically selected at each time instance in both the past and future observations. In other words the developed model has the capability to allow the "birth" of new vehicles or "death" of existing vehicles which makes it applicable in any evolving traffic scene. Moreover, considering a human driver's thinking process, an auxiliary network structure was developed where the model first predicts the future manoeuvre, then uses the predicted manoeuvre to predict the future motion. and then uses both the predicted manoeuvre and the motion to predict the future trajectory. This model has been evaluated using both the publicly available benchmark datasets as well as our newly collected ego vehicle perspective radar dataset.

4.1 Introduction

In a variety of traffic scenarios, it is necessary to interpret the actions and predict the future manoeuvres and trajectories of all vehicles within the field of view of the sensing system. This applies both to traffic monitoring, in which a static sensor may survey a sub-section of the traffic network, or to an autonomous or assisted driving scenario in which the goal is to predict the manoeuvres of surrounding vehicles. In congested traffic scenes, the movement of any 'target' vehicle (TV) is strongly dependant on the road layout and the movement of the surrounding vehicles (SVs),

whose movement further depends on their SVs. Existing models have problems with an increasing, and a variable number of surrounding traffic agents. This forces users to consider only a small, fixed number of SVs with respect to the TV for which the prediction is being performed [119]. Researchers, e.g. [100], [119], tend to consider the same set of SVs with respect to the TV throughout a complete sequence. This is not appropriate for a sequence of any length in which the different vehicles move in different directions or at different speeds, because the closest vehicles at the start of the sequence may have moved much further away.

This chapter discusses new work on behaviour prediction using a manoeuvre-motion-orientated Long Short Term Memory (LSTM) based encoder-decoder architecture. This anticipates the future manoeuvres and positions of all the vehicles in the scene given several seconds of historical observations and associated map features. Unlike existing encoder-decoder architectures, the SV information is incorporated in both the encoder and decoder to allow dynamic selection and updates of the most effective SV movements, boosting the accuracy for longer term prediction. To update the entire scenario after every decoder prediction, each vehicle is considered concurrently as the TV, identifying the most influential SVs with respect to that TV as opposed to the use of predetermined TV and SV data.

The proposed LSTM methodology is applicable to any sensing mode, but an automotive radar was considered as the sensor of interest, and to that end the SVs within a tractable dynamic range of 5-100 metres of the TV were only considered to allow sensible behaviour prediction and time to act. In addition to using established benchmarks, the developed behavior analysis technique was also applied to a newly collected dataset using a 76-77GHz automotive radar mounted on a small van [143]. In most of the recent work, radar was used jointly with other sensors, e.g. camera, LiDAR, and even a V2V communication system, to perform trajectory prediction and avoid any potential collisions. But as shown in Figure A.2 in bad weather (mostly foggy) or poor lighting a reliance on fusion of radar with other sensor modules can lead to a significant loss of performance, and also a strong dependency on the V2V communication can lead to dangerous accidents during dropped connections. Hence, a vehicle detection and prediction system based solely on radar sensing is crucial for level 4 and level 5 autonomy in all conditions, and behaviour prediction is the main focus of this thesis. The main contributions of this chapter are:

- A novel LSTM architecture that predicts vehicle trajectories and manoeuvres with updates from both the historical (known) and predicted manoeuvres as well as motion .

- An approach that reconsiders the entire current scene after each recursive step

in both the encoder (observation-sequence) and decoder (prediction-sequence). This dynamically re-identifies and selects the nearest, and hence most influential SVs at each time step.

- Evaluation of the presented approach against the state-of-the-art using benchmark datasets US junction Lankershim [25], US Highway I-80 [24], US Highway US-101 [23] and the newly collected radar dataset [143] acquired by a 76-77GHz automotive radar. This shows comparable performance to the state-of-the-art and its probable effectiveness to be deployed in an on-board sensing scenario.

4.2 System Model

The problem addressed in this chapter is as follows: given the past observations of all the vehicles in the scene along with various map based features, the task is to predict the future manoeuvre, velocity and trajectory of all those vehicles. Similar to the previous chapter, due to the fixed input size, only the vehicles appearing at the start of the observation sequence will be considered for prediction. The proposed technique in this chapter can be used for both highways as well as for intersections but only after retraining the model. The main reason behind this is the strong dependency on various scene specific map based features. For example distance from the nearest junction will play a very crucial role during trajectory prediction at an intersection whereas in the highways this will become irrelevant. This is why depending on the current context the input feature vector structure needs to be changed and hence the necessity of retraining the model specially when it is trained with only highway data and then tested with intersection data or vice versa.

4.3 Problem Formulation

4.3.1 Problem Statement

A vehicle trajectory is represented as a sequence of points, each of which is associated with a unique time instance $[(x,y)_{t-h}, (x,y)_{t-h+1}, ..., (x,y)_t)]$ where h is the length of the past observation sequence and t is the current time. For N vehicles in a scene, as shown in Figure 4.1, the trajectories, T, are represented as a 2D matrix, where the i'th row holds the trajectory of the i'th vehicle:

$$T = \begin{bmatrix} (x,y)^1_{t-h} & (x,y)^1_{t-h+1} & \dots & (x,y)^1_t \\ (x,y^2_{t-h} & (x,y)^2_{t-h+1} & \dots & (x,y)^2_t \\ \dots & \dots & \dots & \dots \\ (x,y)^N_{t-h} & (x,y)_{t-h+1})^N & \dots & (x,y)^N_t \end{bmatrix}$$

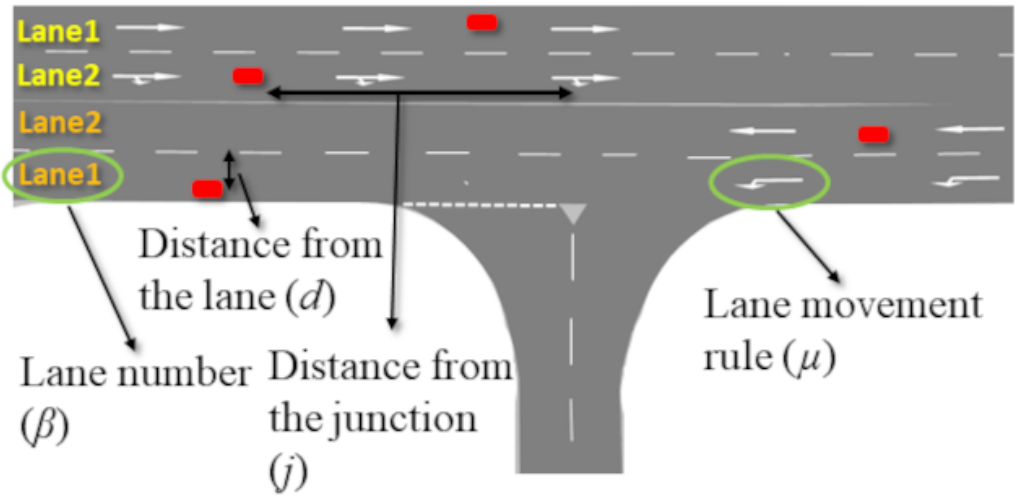


Figure 4.1: Traffic scene showing several vehicles and road network features.

The goal is to predict the manoeuvre class ($\mathbf{m}$), velocity ($\mathbf{v}$) and position ($\mathbf{pos}$) for all vehicles for the next F time instances given T and associated map features, also shown in Figure 4.1. A restricted set of three manoeuvre classes, left turn, right turn and straight on were considered, which are consistent with the data collected and comparable to other work, although this is clearly not exhaustive. The predicted entities are represented as,

$$\begin{bmatrix} (\mathbf{m},\mathbf{v},\mathbf{pos})^1_{t+1} & (\mathbf{m},\mathbf{v},\mathbf{pos})^1_{t+2} & \dots & (\mathbf{m},\mathbf{v},\mathbf{pos})^1_{t+F} \\ (\mathbf{m},\mathbf{v},\mathbf{pos})^2_{t+1} & (\mathbf{m},\mathbf{v},\mathbf{pos})^2_{t+2} & \dots & (\mathbf{m},\mathbf{v},\mathbf{pos})^2_{t+F} \\ \dots & \dots & \dots & \dots \\ (\mathbf{m},\mathbf{v},\mathbf{pos})^N_{t+1} & (\mathbf{m},\mathbf{v},\mathbf{pos})^N_{t+2} & \dots & (\mathbf{m},\mathbf{v},\mathbf{pos})^N_{t+F} \end{bmatrix}$$

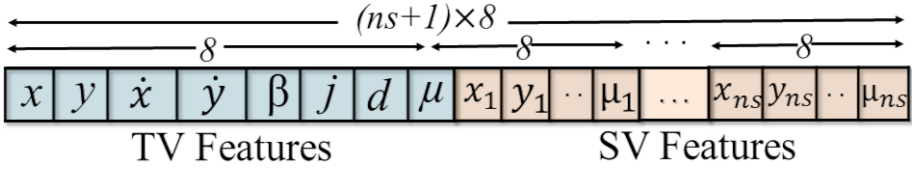
where $\mathbf{m} = [P^s, P^l, P^r]$ is the probability of performing straight, left and right turn manoeuvres respectively, velocity, $\mathbf{v} = [\dot{x}, \dot{y}]$, and position, $\mathbf{pos} = [x, y]$.

4.3.2 Feature Vector Design

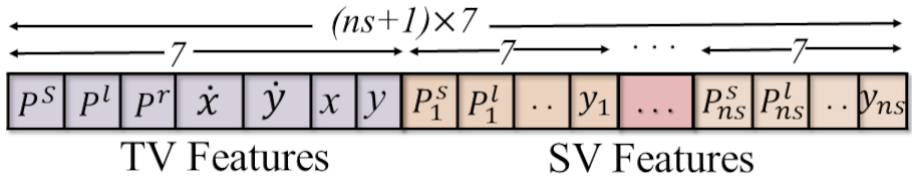
Our feature set includes TV and SV interaction, road-map features and traffic rules.

TV-SV Interaction

To capture the interaction of vehicles the position (x, y), i.e. the front centre of the vehicle (see Figure 4.3) and velocity $(\dot{x}, \dot{y})$ of the TV and the ns nearest SVs at each frame in the last h frames were used. To make the model independent of any specific map relative instead of absolute position was used. The position of the target vehicle, $(x, y)^{TV}_{t-h}$, at the beginning of the historical sequence is considered as the origin for whole sequence. For velocities the absolute values for both the TV and SVs were considered instead of the relative values. This is to differentiate between



(a) Feature vector of the encoder input, $\mathbf{O}_t$, where x, y, $\dot{x}$, $\dot{y}$, l, j, d and μ are the features associated with the TV. x_k,y_k ... μ_k are the features associated with the SVs $k \in \{1, 2, ..., ns\}$.



(b) Feature vector of the decoder input $\mathbf{D}_t$, where P^s, P^l, P^r, $\dot{x}$, $\dot{y}$, x and y are the predicted entities associated with the TV. P^s_k, P^l_k ... y_k are the predicted entities associated with the SVs $k \in \{1, 2, ..., ns\}$.

Figure 4.2: Detailed structure of the encoder and decoder input feature vectors.

high and low speed driving.

Road Map Features

Map-based features strongly influence the future movement of vehicles. The considered features are : 1) distance from the closest junction (j), 2) lane-based localization, i.e. the current lane number of the vehicle (β) and 3) the lateral distance within that lane (d).

Traffic Rule Features

Future movement of vehicles in a multi-lane road is strongly influenced by the rules (μ) assigned to each lane. For instance, at a junction some lanes are marked for a single specific manoeuvre, e.g. a left or right turn. Hence, a vehicle in a designated lane should (but may not) make a proscribed manoeuvre. Similarly, for motorways the outermost lane is for overtaking only. A vehicle in that lane has a higher chance of moving back to the inside lane even if the outermost lane is empty. However, accurate lane based localization of vehicles using a GIS is hard to perform in real time and prone to error.

During prediction, each vehicle is considered as the TV once and the feature vector is created by populating features associated with that vehicle, followed by the SVs. The feature vector structures, $\mathbf{O}_t$ and $\mathbf{D}_t$, at time t are shown in Figure 4.2(a) and Figure 4.2(b), respectively. To keep the input data dimensions fixed, ns was kept constant. If there are more than ns SVs present in the current scene,

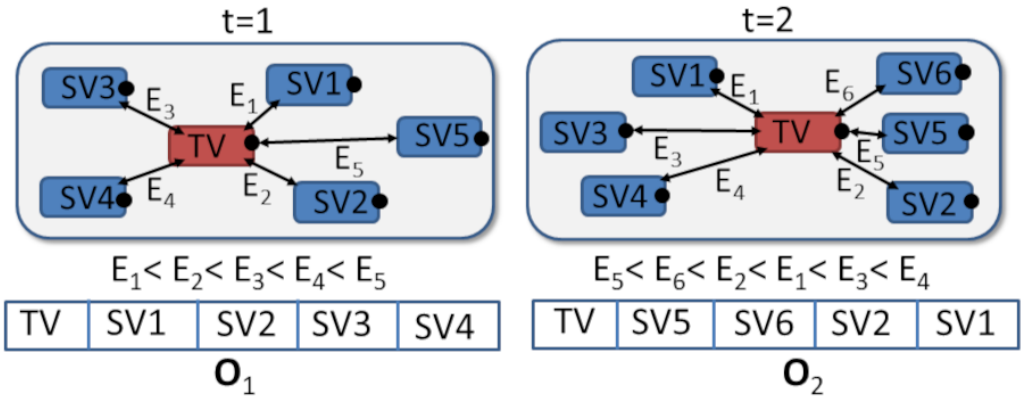


Figure 4.3: Dynamic SV selection at each timestamp based on the Euclidean distance from the TV. E_k is the Euclidean distance between TV and SV_k and $\mathbf{O}_t$ is the input feature vector at time t. The black dots at the front-centre of each vehicle show their point mass locations used for feature vector creation and Euclidean distance calculation between individual vehicles.

Euclidean distance from the TV was used to identify the nearest ns vehicles.

Figure 4.3 illustrates the process of selection of surrounding vehicles. When considering a single TV, the closest (ordered by Euclidean distance) SVs are selected for the feature vectors input to the encoder and decoder at each timestamp, as shown in Figure 4.2. Thus, in Figure 4.3, at $t = 1$ the SVs are the ordered set $\{1, 2, 3, 4\}$, at $t = 2$ the SVs are $\{5, 6, 2, 1\}$. This strategy is in contrast to the common alternative of fixing the SVs at the beginning of the observation sequence, in which case they might move out of the sphere of influence on the TV yet still be encoded. While performing this dynamic selection process, vehicles from both sides of the road were considered as the SV depending on their distance from the target vehicle. In the case of multi lane highways it might be redundant as vehicles travelling on the opposite side of road with respect to target vehicle should never appear on the same side but in case of single carriageway or at an un-signalled intersection this is not true. At an un-signalled intersection, before making a turn it is important to consider oncoming vehicles on the opposite side of the road trying to pass the intersection or making a different turn. Hence, filtering out those SVs would make the model applicable only in multi-lane highway scenarios and not in other locations. Moreover, these SVs were populated not just with their positions but also with their associated map features, e.g. in which lane the vehicle is travelling or how far that vehicle is from an intersection. During the training process these associated features helped the model to understand how individual SVs, even when they are not on the same side of the road, can influence a TV's future movement. When there are less than ns surrounding vehicles present zero padding was used to keep the input dimension fixed.

4.4 Comparison with the OGM Approach

As explained in the previous chapter, in the OGM based approach, if there are N vehicles in the scene the technique creates N sequences of OGMs of dimension 1024, 128 and 30. Where 1024 and 128 are the number of rows and columns of each OGM, respectively. 30 is the number of frames in each past observation sequence. This made the final size of each input for each vehicle equal to 3932160 (1024∗128∗30). In the case of the feature vector based approach proposed in this chapter the input size for each vehicle is only $(4+1)*8*30 = 1200$, where 4 is the number of surrounding vehicles considered, 1 is the target vehicle for which the prediction is performed and 8 is the number of features associated with each vehicle, as explained in the previous section. As shown below, this change of approach led to an almost 200% reduction in the input size

$$\frac{|3932160 - 1200|}{\frac{3932160+1200}{2}} = 199.878\% \tag{4.1}$$

Moreover, compared to the OGM based approach, the feature vector based technique proposed in this chapter also reduced the size of the intermediate entities created to perform the interactive prediction stage. In the case of the OGM based approach, after predicting the future position of each vehicle for the current prediction time instance, the proposed technique recreates the individual OGMs of dimension 1024*128=131072 for each vehicle before predicting the next time instance. In the case of the feature vector approach explained in this chapter only the decoder input of shape (4+1)*7 = 35 is created, where 4 is the number of surrounding vehicles considered, 1 is the target vehicle and 7 is the number of predicted entities, which includes 3 predicted manoeuvres (left turn, right turn, and straight or left lane change, right lane change and straight on), 2 predicted velocities ((v_x, y_x)) and 2 predicted positions (x, y). As shown below, in this case the technique used in this chapter leads to an almost 200% size reduction during the intermediate prediction stage.

$$\frac{|131072 - 35|}{\frac{131072+35}{2}} = 199.893\% \tag{4.2}$$

Reducing the size of the model input as well as the intermediate entities created to perform the interactive prediction stage, makes the current technique more efficient in both time and memory consumption.

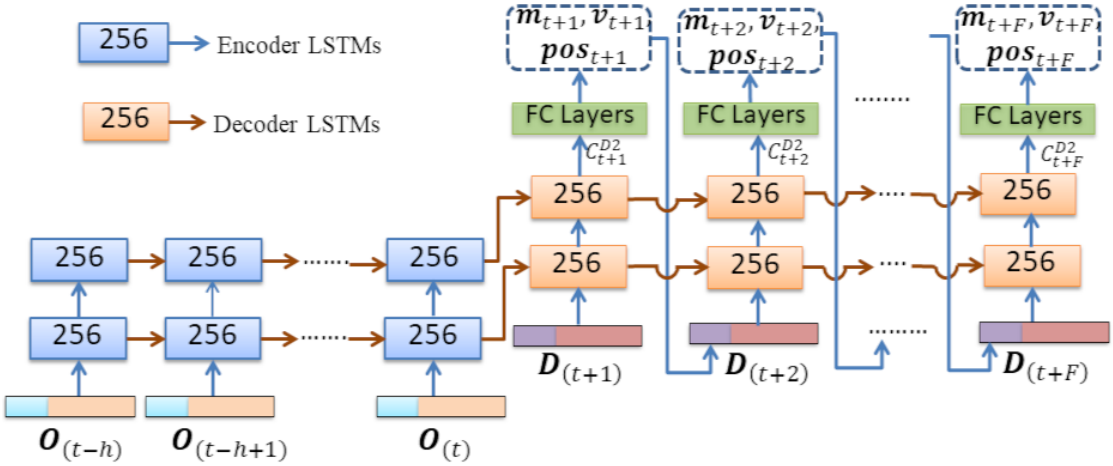


Figure 4.4: The proposed LSTM based encoder-decoder Structure, using the LSTM cell with a 256 neuron count. An example LSTM cell is shown in Figure 2.4. FC Layers are the fully-connected layers shown in Figure 4.5. $\mathbf{O}_{(t-h)}$ to $\mathbf{O}_{(t)}$ are the encoder-inputs shown in Figure 4.2(a) consisting of the TV's features (sky-blue) and nearest ns SV's features (orange). $\mathbf{D}_{(t+1)}$ to $\mathbf{D}_{(t+F)}$ are the decoder-inputs shown in Figure 4.2(b) consisting of the predicted entities of the TV (purple) and the predicted entities of nearest ns SVs (red).

4.5 Future Trajectory Prediction using a LSTM Structure

Our full network architecture consists of an encoder-decoder structure, with associated manoeuvre learning, velocity learning and trajectory learning modules shown in Figs. 4.4 and 4.5.

4.5.1 Encoder-Decoder Module

The encoder processes the underlying, input temporal information and passes it to the decoder. As shown in Figure 4.4, the encoder is a stacked-LSTM network [40] consisting of two LSTM layers which take $\mathbf{O}_{t-h,t}$ as input. Both LSTM layer states were initialized randomly. In most recent encoder-decoder based trajectory prediction models, the decoder is responsible for capturing the encoded state from the encoder and produces future positions using only the TV's past positions, not using any information regarding the SVs. This makes the decoder outdated as the future horizon extends. This problem was addressed by adding the nearest ns SV's previous manoeuvres ($\mathbf{m}_t$), velocities ($\mathbf{v}_t$) and positions ($\mathbf{pos}_t$) into the decoder-input vector $\mathbf{D}_{t+1}$, while predicting the state vector at $t+1$,

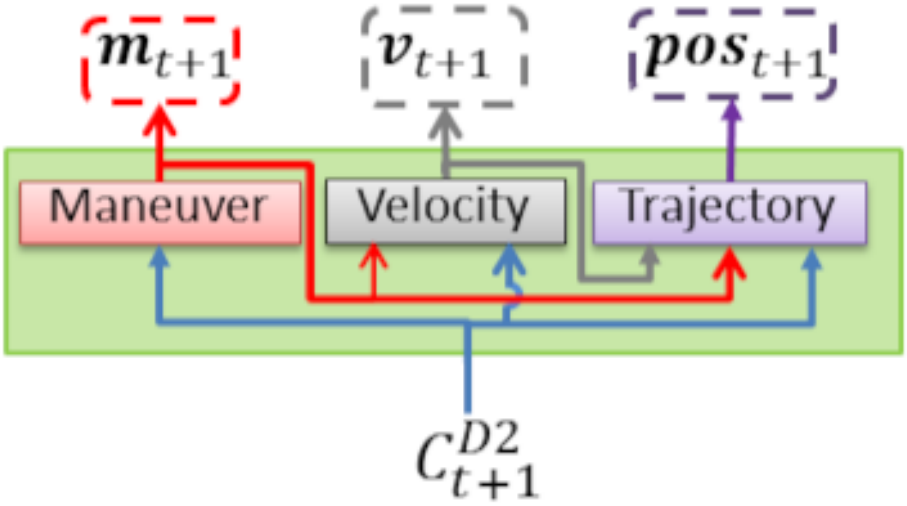


Figure 4.5: Proposed fully-connected (FC) layer structure. The manoeuvre, velocity and trajectory sub-structures consist of multiple FC layers. The blue arrows show the LSTM cell output from the decoder. The red arrows show the predicted manoeuvre input to the velocity and trajectory sections, The grey arrows show the predicted velocity into the trajectory section. Finally the purple arrow shows the predicted position.

$$\mathbf{D}_{t+1} = [\mathbf{m}_t, \mathbf{v}_t, \mathbf{pos}_t, \mathbf{m}_t^1, \mathbf{v}_t^1, \mathbf{pos}_t^1, ..., \mathbf{m}_t^{ns}, \mathbf{v}_t^{ns}, \mathbf{pos}_t^{ns}] \tag{4.3}$$

where, $\mathbf{m}_t$, $\mathbf{v}_t$, $\mathbf{pos}_t$ are the predicted entities of the TV and $\mathbf{m}^k$, $\mathbf{v}^k$, $\mathbf{pos}^k$ are the predicted entities of the SV $k \in \{1, 2, .., ns\}$.

The decoder-input feature vector $\mathbf{D}_{t+1}$ is shown in Figure 4.2(b). The decoder also consists of two stacked LSTM layers [40] which take $\mathbf{D}_{t+1}$ as input. To feed the encoded states the last memory states from the first and second LSTM layers in the encoder were passed to the decoder to initialize the states of its first and second LSTM layers respectively. As the decoders are mainly responsible for producing the sequence output it is safe to ignore the state outputs from the LSTM layer and only consider the final cell output (C^{D2}_{t+1}) for the succeeding subsections. The intermediate cell output (C^{D1}_{t+1}) from the first LSTM layer has already been passed to second LSTM layer.

4.5.2 Manoeuvre Learning

The manoeuvre learning module is responsible for prediction of the manoeuvre class $\mathbf{m}$ for each time instance in the sequence, then used as a prior for velocity and trajectory learning. As shown in Figure 4.5, this module takes the decoder output C^{D2}_{t+1} as input and predicts the TV manoeuvre $\mathbf{m}_{t+1}$ for the next time steps. The

module consists of fully connected layers with neuron counts of $512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 3$ (this is the $\mathbf{m}_{t+1}$). The last layer's neuron count is consistent with the 3 manoeuvre classes (P^s, P^l and P^r). An Exponential Linear Unit (ELU) with $\alpha = 0.1$ was used as the activation function for all but the last layer. The last layer has Softmax [41] activation to produce the probability of each manoeuvre. This normalizes the output vector as a distribution consisting of three probabilities (P^s, P^l, P^r) proportional to the exponential of each cell value in v_m,

$$[P^s, P^l, P^r] = \left[\frac{exp^s}{\sum_{n \in v_m} exp^n}, \frac{exp^l}{\sum_{n \in v_m} exp^n}, \frac{exp^r}{\sum_{n \in v_m} exp^n}\right]$$

4.5.3 Velocity Learning

Due to the strong dependency of future velocity on the planned manoeuvre, this module takes both C^{D2}_{t+1} and $\mathbf{m}_{t+1}$ as input and predicts both velocity components $(\dot{x}, \dot{y})$ as shown in Figure 4.5. C^{D2}_{t+1} and $\mathbf{m}_{t+1}$ are concatenated before being passed to the fully connected layers. The neuron counts of the layers in this module are $512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 2$. Due to the presence of two motion components, $(\dot{x}, \dot{y})_{t+1}$, the last layer neuron count is two. An ELU with $\alpha = 0.1$ was used as the activation function for all the layers except the last one. As the underlying task is regression, "linear" activation was used in the last layer to predict continuous $\dot{x}$ and $\dot{y}$.

4.5.4 Trajectory Learning

As shown in Figure 4.4, the final section predicts future position $(x, y)_{t+1}$, given C^{D2}_{t+1}, $\mathbf{m}_{t+1}$, $\dot{x}$ and $\dot{y}$. These were concatenated as the intermediate input vector $[C^{D2}_{t+1}, P^s_{t+1}, P^l_{t+1}, P^r_{t+1}, \dot{x}_{t+1}, \dot{y}_{t+1}]$ before feeding to the fully connected layers. The neuron counts of each layer are $1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 2$. There are two neurons in the last layer, corresponding to x_{t+1} and y_{t+1}. Again, the last layer has a "linear" activation, and an ELU with $\alpha = 0.1$ for all other layers.

4.5.5 Loss Functions

Our proposed architecture predicts three entities (manoeuvre, velocity and position), so three distinct loss functions were used.

Manoeuvre Loss

The three ground truth manoeuvre class labels in the training sample are formatted as one hot vector where a probability value of 1 was assigned to the true manoeuvre

class and 0 to the rest. This representation makes it easier to compute the categorical cross entropy loss [169], the negative log-likelihood between the predicted and true manoeuvre classes over all the training samples

$$\mathcal{L}_m = -\sum_{i \in v_m} (P^i \times \log(\hat{P}^i)) \tag{4.4}$$

where $\mathcal{L}_m$ is the manoeuvre loss, $(\hat{P}^s, \hat{P}^l, \hat{P}^r)$ and (P^s, P^l, P^r) are the predicted and ground-truth manoeuvre class probabilities, respectively.

Velocity Loss

The loss of Eq. (4.5) measures the difference between the ground-truth and the predicted velocity components:

$$\mathcal{L}_v = \log(\cosh(\dot{x} - \hat{\dot{x}})) + \log(\cosh(\dot{y} - \hat{\dot{y}})) \tag{4.5}$$

where $\mathcal{L}_v$ is the computed velocity loss, $(\dot{x}, \dot{y})$ and $(\hat{\dot{x}}, \hat{\dot{y}})$ are the ground-truth and predicted velocity components respectively. The robustness achieved due to the combined effect of the logarithm and *cosh* functions makes the training stable during the occasional wildly incorrect prediction [169].

Trajectory Loss

Trajectory learning was achieved by minimizing the Euclidean distance between the predicted and ground truth positions over all the samples:

$$\mathcal{L}_{traj} = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2} \tag{4.6}$$

where $\mathcal{L}_{traj}$ is the computed trajectory loss, (x,y) and $(\hat{x}, \hat{y})$ are the ground truth and predicted positions, respectively.

4.5.6 Interactive Prediction

The proposed network predicts **m**, **v** and **pos** for all vehicles in the scene for the next F frames. The prediction has two stages. In the first stage, N encoder-input feature vectors $\mathbf{O}_{t-h,t}$ were created considering each vehicle as a TV once and its ns nearest vehicles as SVs, of N cars present in the scene. As future vehicle positions are unknown, at time t all three entities (**m**, **v** and **pos**) are predicted for all the vehicles at the next time instance $(t+1)$, and these predicted entities update each

decoder-input before predicting for $t+2$, and so on till $t+F$. Similar to the encoder-input structure, the predicted entities of the TV followed by the nearest ns SV's were populated in the decoder-input shown in Figure 4.2(b). The nearest and most influential SVs were highlighted by calculating the Euclidean distance between the current predicted position of the TV and the predicted positions of the other vehicles in the scene. This made it possible to reconsider any new vehicle ("birth"), or drop any existing vehicle ("death") that moves in or out of the considered neighbourhood. The detailed steps involved in input feature vector creation and interactive sequential prediction are explained in Algorithm 3 and 4, respectively, using a pseudo-code format.

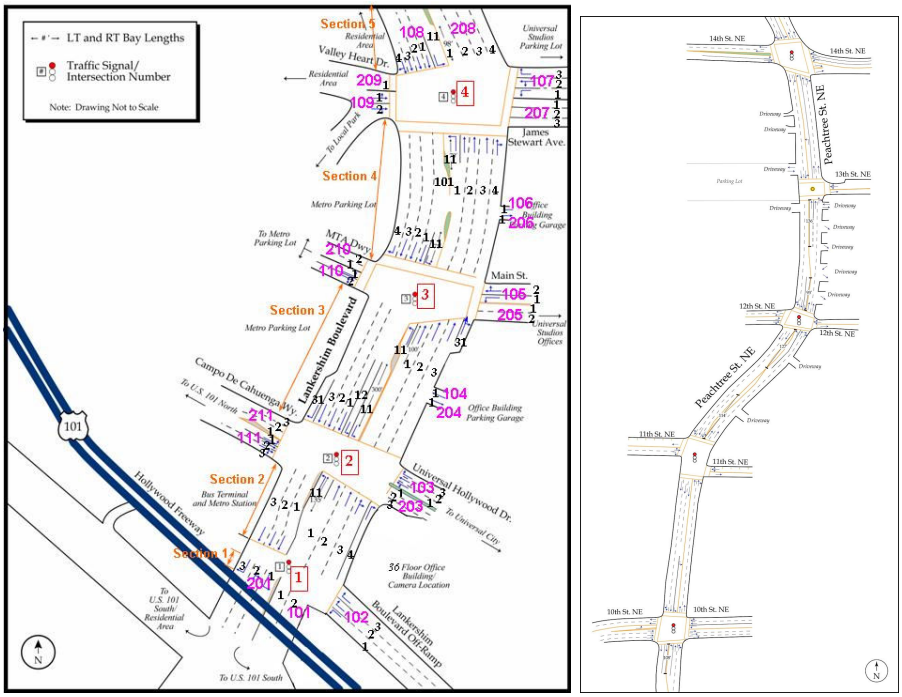
During prediction of future movement of a particular TV at $t+f$, $(f \in 2, 3, ..., F)$ where f is the current prediction instance, the predicted entities of the most influential SVs at $t+f-1$ are also considered in the decoder-input. Hence, this feedback scheme updates the scene information repeatedly to incorporate the potential future interaction of SVs more effectively. In the decoder, the SV's information is shared with the TV by selecting and appending a fixed number (ns) of the most influential vehicle features in the decoder-input instead of directly connecting individual LSTM cells [74] or nodes [100]. This makes the proposed technique scalable to any number of vehicles present in the scene.

4.6 Experimental Procedure and Use of Data

Although the developed methodology is targeted at data arising from automotive radars, there was no suitable radar dataset or public benchmark available at the time. Hence, it was evaluated with a newly collected radar dataset [143]. To further compare the developed approach with the state-of-the-art, it was also evaluated using the two publicly available datasets, the Lankershim [25] and combined I-80/US-101 highway datasets [24] [23] collected by video cameras.

4.6.1 Evaluation on Benchmark Dataset

The NGSIM dataset explained in Section 3.6.1 also includes two intersection datasets, Lankershim [25] and Peachtree [26]. Lankershim [25] covers a 500 meters (1600 feet) long, 3-4 lane wide section on Lankershim Boulevard in Los Angeles (see Figure 4.6(a)) which includes 4 intersections. Peachtree covers a 640 meter (2100 feet) long, 2-3 lane wide section of Peachtree street in Atlanta which includes 5 intersections (see Figure 4.6(b)). Each data point in these datasets has all the data fields mentioned in Section 3.6.1, plus the O_Zone (Origin Zone, from where the vehicle entered the area) and D_Zone (Destination zone, from where the vehicle exited



(a) Lankershim Junction [25] (b) Peachtree Street [26]

Figure 4.6: Bird's Eye View (BEV) of two intersection locations in NGSIM dataset, i.e. Lankershim Boulevard and Peachtree Street, Atlanta. Each sub figure consists of a satellite view of the observation area and a schematic diagram with lane counts

the area), Int_ID and Section_ID (intersection or Section ID at which the vehicle is currently travelling), Direction (in which direction the vehicle is currently travelling, i.e. East-Bound, North-Bound, West-Bound, or South-Bound) and Movement (movement rule associated to the current lane). Although, the Lankershim data [25] is more interesting with several intersections, most previous methods have been evaluated on the combined NGSIM US-101 [23] and I-80 [24] dataset. Therefore, to conduct a fair comparison, the developed model was evaluated on these samples first and then on the intersection data.

In both cases, i.e. I-80/US-101 highway and Lankershim intersection data, the locations estimated through the global state plane co-ordinate system were used for trajectory generation instead of the specific local co-ordinate system to make it more generic. The instantaneous speeds $(\dot{x}, \dot{y})$ at each frame were estimated from successive frames instead of using V_Vel available directly from the dataset. The lane number (β) and lane movement rule (μ) were populated directly from the dataset. Details about both these datasets are given in Section 3.6.1. To label each trajectory with a single manoeuvre the vehicle's origin zone (101-111, Figure 4.7,) and destination zone (201-211, Figure 4.7) was used. For example, if a vehicle enters from zone 101 and exits through zone 203, this means the vehicle has performed a right turn. In addition to this a combination of the current section and driving

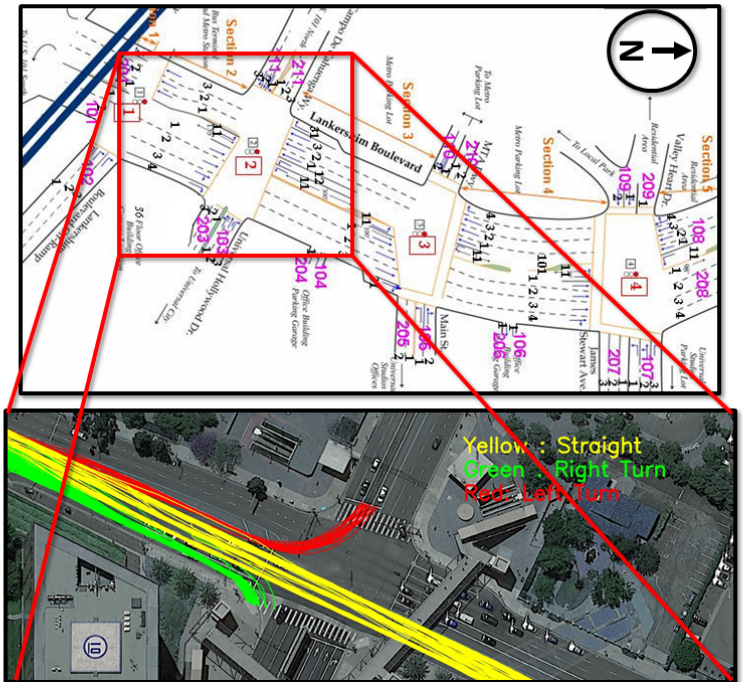


Figure 4.7: Schematic diagram with junction and section locations of the NGSIM [25] evaluation area, and a satellite view [111] with a few sample trajectories on one selected junction, where green, yellow and red indicate vehicles performing straight, right and left manoeuvres at that junction, respectively.

direction were used to identify the nearest junction and calculate the additional feature "distance from the junction". Among 2413 detected vehicles only 1036 vehicles performed either a left or right turn manoeuvre at any intersection. The number of vehicles associated with each manoeuvre is given in Table 4.1.

4.6.2 Evaluation on the "Radiate" Dataset

As mentioned earlier the main goal of this thesis is to perform surrounding vehicle behaviour prediction using ego-vehicle perspective realistic data collected with on-board sensors. This is why the developed model was also evaluated on a newly collected radar dataset "Radiate". It includes of data collected by four different sensor modalities, a 77 GHz Navtech radar, a 32 channel velodyne LiDAR, a ZED stereo camera and an Advance Navigation GPS/IMU tool kit. The data collection software was developed using the Robot Operating System (ROS) [125] framework which collects and stores the data in a "rosbag". Later, individual sensor data is extracted along with their respective timestamps. The complete dataset consists of 5 hours of radar images along with the additional data collected through the other sensors. Since radar is the main focus of this thesis, 3 hours worth of radar images are fully annotated; this consists of more than 200K labelled object instances with

Table 4.1: Vehicle distribution for each manoeuvre class in all three junctions, i.e. the Lankershim, Kingussie (T) and Edinburgh (Four-way) junctions.

	Number of Vehicles			
Junction	**Straight**	**Left**	**Right**	**Total**
Lankershim (NGSIM)	1377	548	429	2354
Kingussie (T) (Radiate)	118	–	46	164
Edinburgh (Four-way) (Radiate)	111	26	31	168

8 categories of road actors i.e. car, truck, van, bicycle, motorbike, pedestrian and group of pedestrians. Moreover it also includes a variety of weather conditions, e.g. sun, rain, night, snow, and fog, and various driving scenarios e.g. motorway, urban, parked, and suburban), representing different levels of challenge. Even though the collected data includes both highways and intersections, this chapter focuses mainly on the vehicle-to-vehicle interactions at signal-free intersections where the behaviour prediction task is more challenging. This is because missing traffic infrastructure such as traffic lights or turn specific lanes makes the inter-vehicle negotiation solely dependent on the human driver's behaviour. The model evaluation commenced with a simple "T-junction" scenario (Kingussie), shown in Figure 4.9(a), having mild congestion. Then the evaluation moves to a denser and more complex "Four-way junction" (Edinburgh), shown in Figure 4.9(b). In both these cases the test vehicle was parked near the junction (see Figure 4.8). In some of the recent vehicle trajectory prediction work [29], [180], [30], [183], [31] the lateral or sideways movements of all the vehicles on the main road were maintained along the X-axis and the forward movements were maintained along the Y-axis. Considering this co-ordinate convention and the horizontal main road in front of the parked test vehicle (see Figure 4.8), the inverted co-ordinate system which is vertical x-axis and horizontal y-axis was adopted in case of the "Edinburgh Four-way junction" dataset. Initially, false or missed detections were not considered, rather human annotated bounding boxes (as is indeed the case with the NGSIM data) were used to generate individual vehicle trajectories which were smoothed using particle filters [56] to remove any unwanted vibration introduced during the annotation. Similar to the NGSIM data, the origin and destination zones were used to label each vehicle trajectory against one of the designated manoeuvres. The numbers of vehicles for each manoeuvre class for each junction are shown in Table 4.1. Compared to the NGSIM data, the relatively smaller range of the vehicle sensors allowed it to capture only 30 to 50 frames for each vehicle with a frame rate of 4Hz. Examples of collected and labelled trajectories are shown in Figure 4.9.

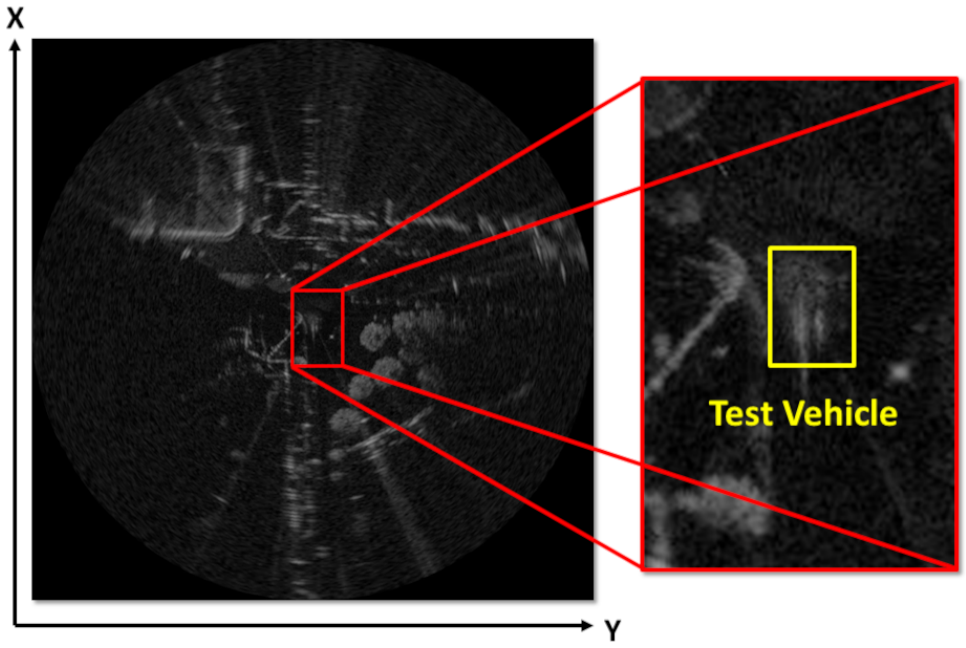
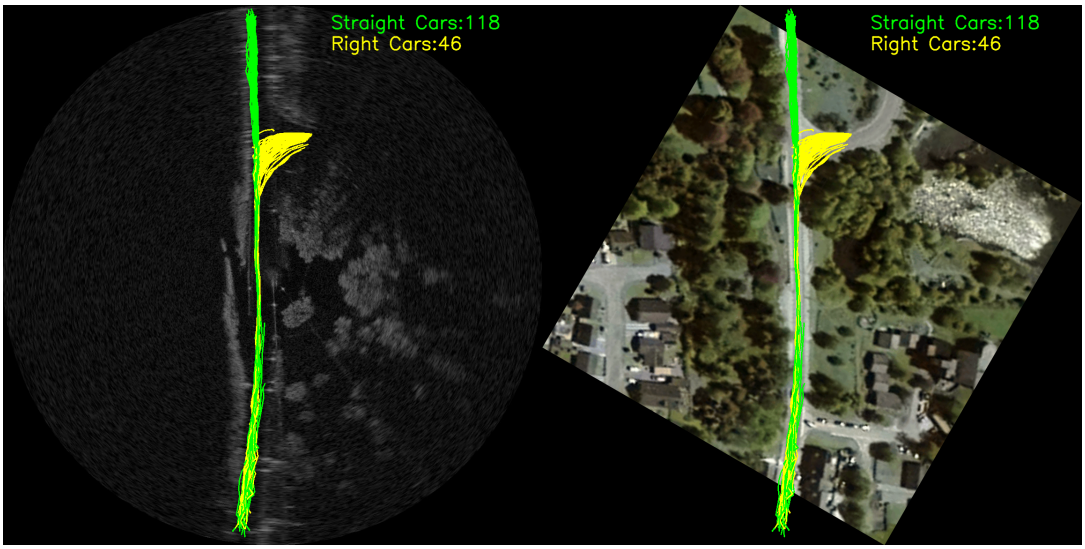


Figure 4.8: A labelled radar image showing the test vehicle parked at a junction. The actors are denoted by boxes which can be of any size and at any angle to the image frame.

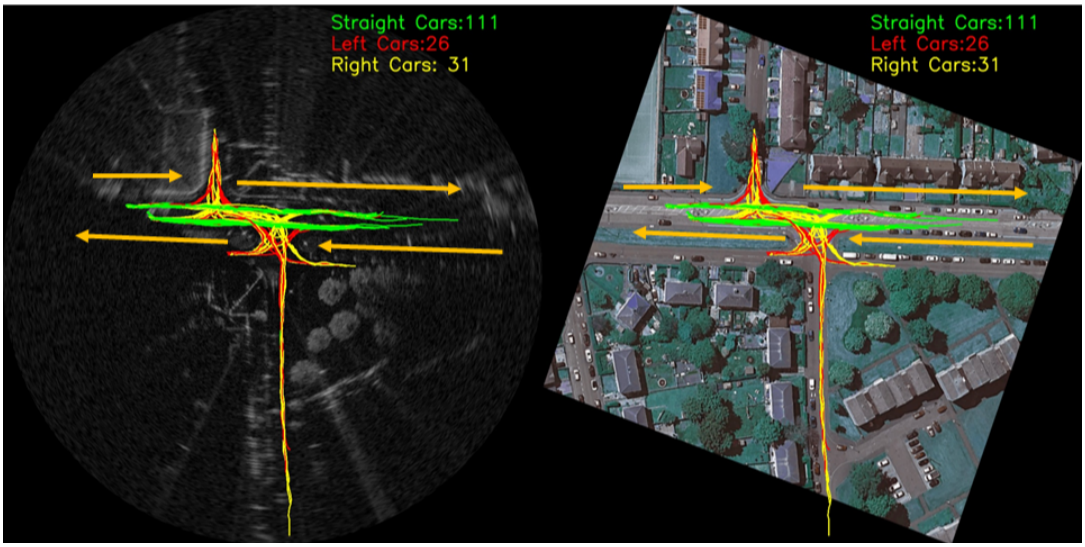
4.6.3 Sample Generation

For efficient sample generation from the Lankershim dataset initially two dictionaries were created i.e. "Vehicle_Dict" and "Frame_Dict" with Vehicle_ID and Frame_ID as the respective keys. Each vehicle in the Vehicle_Dict was selected as the TV once; the current Frame_ID of the TV together with the Frame_Dict was used to identify the SVs present at that time to create the input trajectory matrix, T. The predicted SV's positions were also used in the decoder; future position prediction is only possible for the vehicles whose last $h = 30$ frames positional information is available. Hence, to keep the context consistent between the observation and prediction sequences, only the vehicles present in the last $h = 30$ frames were selected as SVs. Once T was created the Euclidean distance was used (Figure 4.3) within a sliding window to identify the most influential ns SVs at each timestamp.

For example, consider generating samples for one specific TV with SVs $1, 2, 3$. The observation sequence of the first sample is generated using the TV and SV's positions from t_0 to t_{29} for ($h = 30$ frames). The ground truth for that sample is the TV's position from t_{30} to t_{79} for ($F = 50$ frames). For the next sample the time window was shifted one unit forward so that the observation sequence is from t_1 to t_{30}, and the ground truth is the TV's position from t_{31} to t_{80} and so on. The process is repeated for each vehicle as TV. The number of generated samples are given in Table 4.2 where each NGSIM sample has 30 observation plus 50 prediction frames and each Radiate sample has 12 (3 secs with 4 Hz Navtech radar acquisition rate)



(a) Kingussie (T) junction [110]



(b) Edinburgh (four-way) junction [108]

Figure 4.9: The trajectories of each manoeuvre class for two different junctions collected with the test vehicle [143]. Each sub-figure shows the satellite view [108], [110] of the observed junction (right) and its view through the radar system (left). Green, red and yellow lines show the vehicle performing straight-on, left-turn and right-turn manoeuvres, respectively. The yellow arrows indicate the driving directions.

observation plus 20 (5 secs) prediction frames.

4.6.4 Under-sampling Technique to Avoid Data Imbalance

In the Lankershim and US-101/I-80 datasets the number of vehicles travelling straight on is higher than those making a turn or a lane change. This imbalanced data can bias the network. To avoid this the vehicles performing the straight on manoeuvre was under-sampled by selecting less vehicles from this class compared to the turns or lane changes in the training data. Applying under sampling can remove the underlying prior probability information of performing various manoeuvres. For example, if there is a supermarket at the right turn of a junction it is highly likely that more vehicles will perform that turn compared to left turn or straight on. However, this is true only for that specific junction and at a different junction the prior probability can be different depending on where it is in the real world. In order to remove this junction-specific prior probability the under sampling was performed before training the model. In that way the model should better learn how the target vehicle's future manoeuvre, velocity and trajectory is predicated by the relative positions of all the surrounding vehicles and their associated features rather than be predominantly junction-specific.

From TABLE 4.1 the vehicle count performing the straight on manoeuvre in the Lankershim dataset is almost three times the vehicle count performing either a right or left turn. The under-sampling for this dataset was done using a two stage process, where initially initially all the vehicles were categorized under two categories, i.e. "straight" and "turning" vehicles. For example, a vehicle entering zone 108 or 101 and exiting zone 201 or 208 respectively means the vehicle never performed a right or a left turn manoeuvre (see Figure 4.7) and is added to the "straight" vehicle category. In the next stage, while randomly selecting vehicles less were selected from the "straight" category compared to the "turning" vehicle category and then generated samples using the sliding window technique explained in section 4.6.3. This adopted under-sampling of vehicles performing straight manoeuvres led to better precision for both left and right turn manoeuvre classes.

This under-sampling technique was also adopted for the I80/US101 dataset. In the I80/US101 data some of the vehicles never performed a lane change and the rest of the vehicles did it only once in the entire observation area shown in Figure 3.5. The sliding window sample generation technique explained in section 4.6.3 leads to a relatively higher number of samples whose all the frames will be annotated as straight manoeuvre compare to left or right lane change making the manoeuvre distribution imbalanced. In this case the under-sampling was performed by selecting less vehicles of those that never performed a single lane change from the I80/US101 dataset, as

Table 4.2: Number of generated samples using the sliding window technique for the Lankershim, Kingussie (T) and Edinburgh (Four-way) Junctions.

Dataset	Training	Validation	Testing	Total
Lankershim (NGSIM)	307360	80000	77327	464687
I-80/US-101 (NGSIM)	276130	90500	82500	449130
Kingussie (Radiate)	2880	440	282	3602
Edinburgh (Radiate)	3540	550	405	4495

all the frames in the generated samples from those vehicles will be annotated as straight. On the other hand, those vehicles performed at least one lane change generate a combination of samples where each frame can be marked as either a straight or left/right lane change depending on its Lane_ID in the past 20 frames. If the Lane_IDs are the same in the past 20 frames, this means that no lane change happened whereas a change in Lane_ID indicates a lane change. In both I-80 and US-101 data the left most lanes are marked as 1, the second left most lanes are marked as 2 and so on (Figure 3.5). This means that an increase in Lane_ID indicates a right lane change and a decrease in Lane_ID indicates a left lane change.

4.6.5 Training

The three different loss components $\mathcal{L}_m$, $\mathcal{L}_v$ and $\mathcal{L}_{traj}$ have different scales. $\mathcal{L}_m$ refers to a classification task which makes its value range relatively low compared to $\mathcal{L}_v$ or $\mathcal{L}_{traj}$ as these refer to regression tasks. Simple averaging to combine these three losses into one common loss leads to a situation where the manoeuvre learning section remains under fitted due to its lesser contribution. To address this, a weighted sum approach was adopted to combine all three losses.

$$\mathcal{L} = W_m * \mathcal{L}_m + W_v * \mathcal{L}_v + W_{traj} * \mathcal{L}_{traj} \tag{4.7}$$

where $\mathcal{L}$ is the final loss, W_m, W_v and W_{traj} are the weights for $\mathcal{L}_m$, $\mathcal{L}_v$ and $\mathcal{L}_{traj}$ respectively. Empirically, after multiple trials the weights were set to $W_m = 0.42$, $W_v = 0.29$ and $W_{traj} = 0.29$ with "Nadam" [38] as the optimizer because this works better for recurrent tasks. The step decay learning rate starts from 0.001 and reduces by 50 percent after each 2 epochs. The model is implemented in Keras [85] with the TensorFlow backend. The implementation of the proposed technique can be found here.

4.7 Experimental Evaluation

Evaluation of the approach was performed using all the datasets presented in Section 4.6 measuring firstly the manoeuvre classification accuracy and precision, and secondly the accuracy of trajectory prediction. Where possible, a comparison with previous work is also included.

4.7.1 Manoeuvre Classification

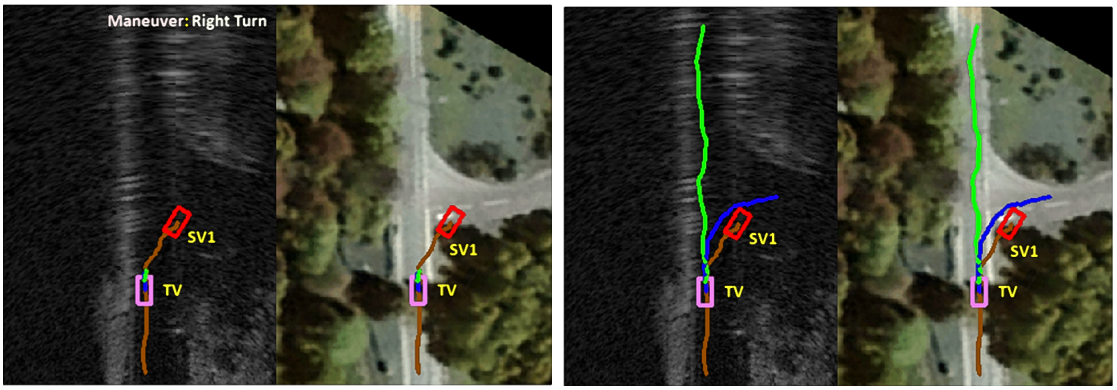
To evaluate the success of manoeuvre classification, the accuracy, A, and precision, P, values were measured by

$$A = \frac{TP + TN}{TP + TN + FP + FN} \quad P = \frac{TP}{TP + FP} \tag{4.8}$$

where TP is a true positive, TN is a true negative, FP is a false positive and FN is a false negative.

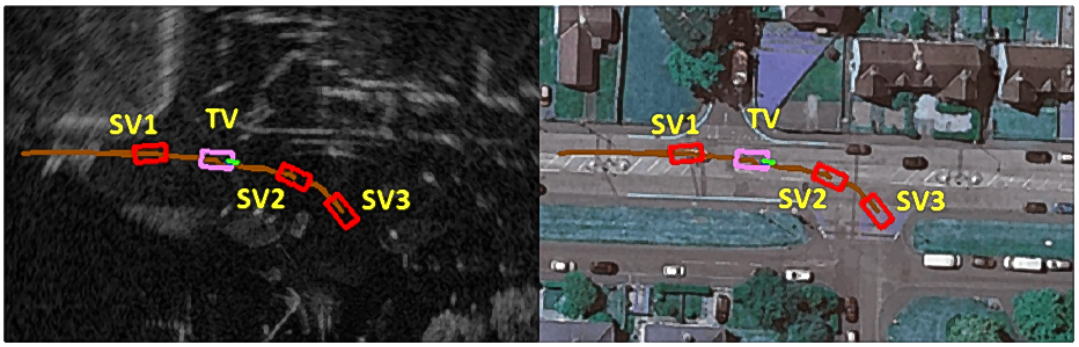
As can be seen in Table 4.3, both metrics are excellent for all four datasets, well in excess of 90% in the majority of cases, dropping to 85.7% for precision in the case of the right turn (Lankershim), and more tellingly to below 60% in the I80/US101 data. For the Lankershim data, the developed technique was compared with Phillips et al. [123] who used a combination of historical, traffic and rule features in an LSTM architecture to predict the manoeuvres. The proposed technique achieved a better accuracy in all three manoeuvre classes when compared to this method, which may be due the inclusion of additional scene information. For the Kingussie and Edinburgh radar data, for which no previous results are available, the estimated RMSE values are comparable. However, in the US-101/I-80 case, left and right manoeuvres are lane changes rather than turns. Hence, there is no parameter defining distance from a junction as none exist, and the velocity profile is liable to be less informative. As these are key parameters in the case of the turns, this may partly explain the lower success in left and right lane changes in that case.

To explain this more clearly the encoded states of all the predicted samples against individual manoeuvre classes were plotted for both intersection and highway datasets. The neuron count of each LSTM cell in the stacked-LSTM encoder is 256. While predicting each sample the encoder will produce four outputs, 2 hidden states and 2 cell outputs as vectors of length 256. As it is very difficult to visualize such a high dimension of the encoded states and cell outputs it was reduced from 256 to 2 using t-distributed stochastic neighbor embedding (t-SNE) [163] and then plotted against their ground truth manoeuvre classes. As shown in Figures. 4.11(a), 4.11(b), 4.11(c) and 4.11(d), the hidden states and the intermediate cell outputs from the encoder created three fairly distinct clusters for the three manoeuvre classes *i.e.*

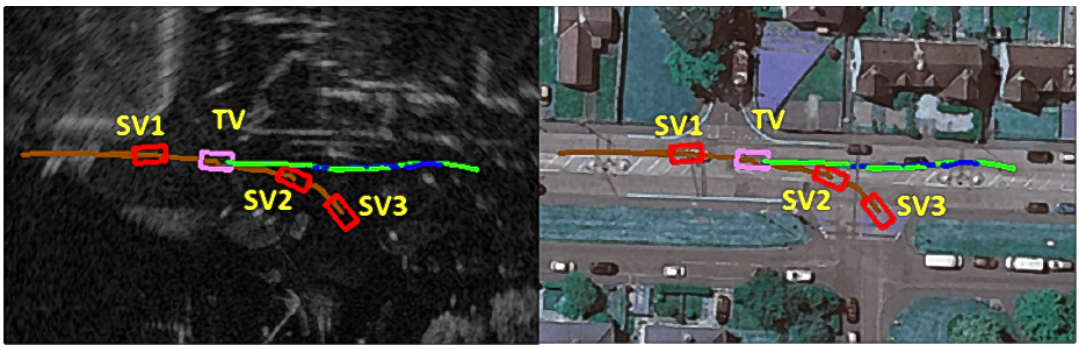


(a) Right turn manoeuvre mis-classification at the Kingussie (T) Junction [110] due to slow velocity profile.

(b) True future trajectory of TV after the Kingussie (T) junction [110].



(c) True straight manoeuvre classification at Edinburgh (Four-way) junction [108] with low velocity.



(d) True future trajectory of TV after the Edinburgh (Four-way) intersection [108].

Figure 4.10: Qualitative analysis of manoeuvre prediction at the Kingussie (T) and Edinburgh (Four-way) junctions. For each subfigure, on the left and right are radar and satellite images [108], [110] respectively. The violet car is the TV, the red cars are the SVs. The future ground truth and predicted trajectories are marked in green and blue respectively. The input trajectories for each vehicle are drawn in brown. A shorter trajectory indicates a slower velocity

right turn, left turn and straight on in case of the Lankershim intersection dataset. These distinct clusters helped the decoder and the rest of the network to achieve

better classification accuracy. In case of the highway (I-80/US-101) dataset, due to the missing key input features, the encoded states and the cell outputs were more scattered (see Figures. 4.12(a), 4.12(b), 4.12(c) and 4.12(d)) leading to a poorer manoeuvre classification accuracy.

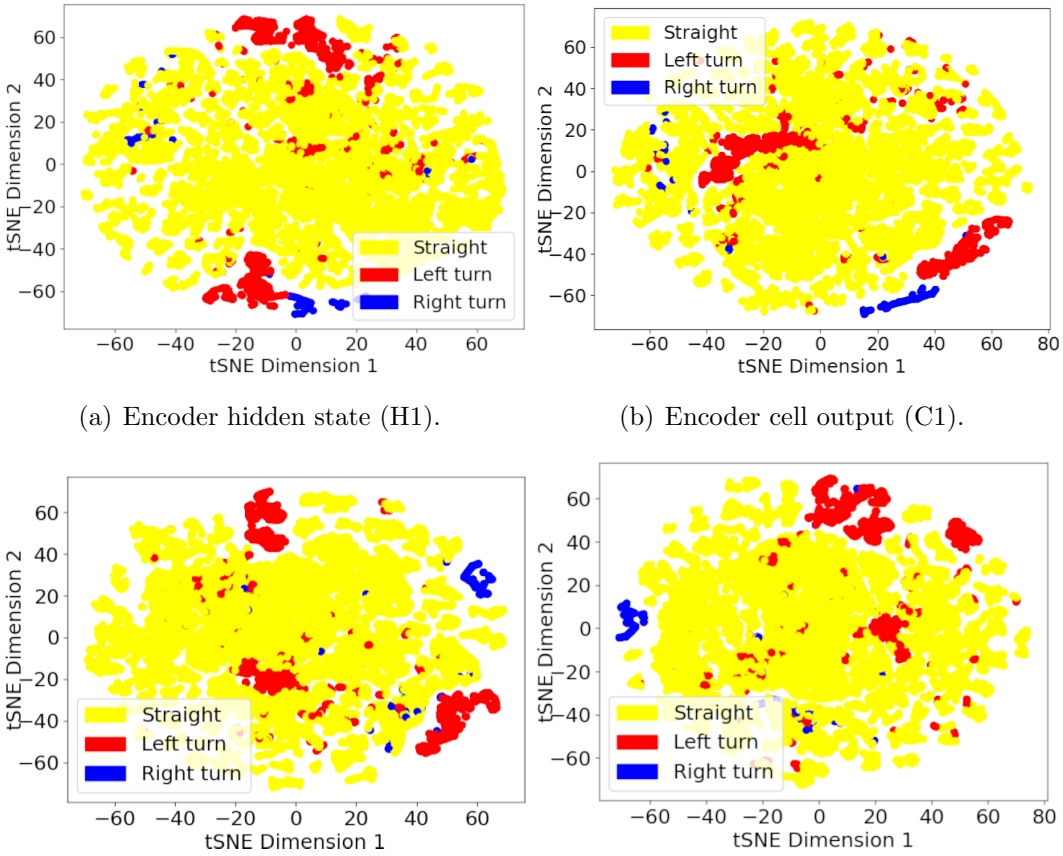
To reinforce this conclusion, distinct velocity profiles between turning and going straight vehicles at the Kingussie (T) junction are also shown in Figure 4.17. There is a noticeable slowing on approach for turning vehicles, so combined with the junction distance variable, this helps the model achieve close to 100% accuracy for manoeuvre classification. However, this is not infallible. Noticeable slowing of a vehicle may be because of the presence of SVs and not because it makes a right turn. As shown in Figure 4.10(a) the model predicts a right turn for the TV as it was close to the junction and had a slowing velocity profile, but this was due to the presence of SV1 as it was performing a safe right turn manoeuvre. Once SV1 completes the right turn the TV accelerates and continues on the straight road, but the previous sub-sequence led to a right turn future trajectory (see Fig 4.10(b)). This is possibly because of the lack of sufficient SV interactions in the training data made it harder for the model to distinguish deceleration before a turn manoeuvre or the presence of a SV.

On the other hand at the Edinburgh (Four-way) junction the velocity profiles of the target vehicles are affected not only by the junction location but also by the presence of surrounding vehicles. Hence, although not as impressive as the simpler T-junction case, the accuracy and precision measures are both high. A true manoeuvre classification for a TV, even though it had a slow velocity profile and was close to the junction, is shown in Figs. 4.10(c) and 4.10(d). Due to the presence of sufficient SV interactions in the training data the model was able to conclude that the slow velocity profile was caused by congestion. Figure 4.10(d) shows the true future trajectory. To better understand why errors occur, class based normalized confusion matrices are presented in tables 4.4 to 4.7. A demo of this work can be found here.

4.7.2 Trajectory Prediction

Assessment on the NGSIM I-80 and US-101 Datasets

In contrast to manuever prediction, several authors have used the NGSIM I-80 [24] and US-101 [23] datasets for trajectory prediction and this gives a point of comparison with the developed approach. The Root Mean Square Error (RMSE) between the predicted and ground truth trajectories is used as a metric and compared with the following approaches:



(a) Encoder hidden state (H1).

(b) Encoder cell output (C1).

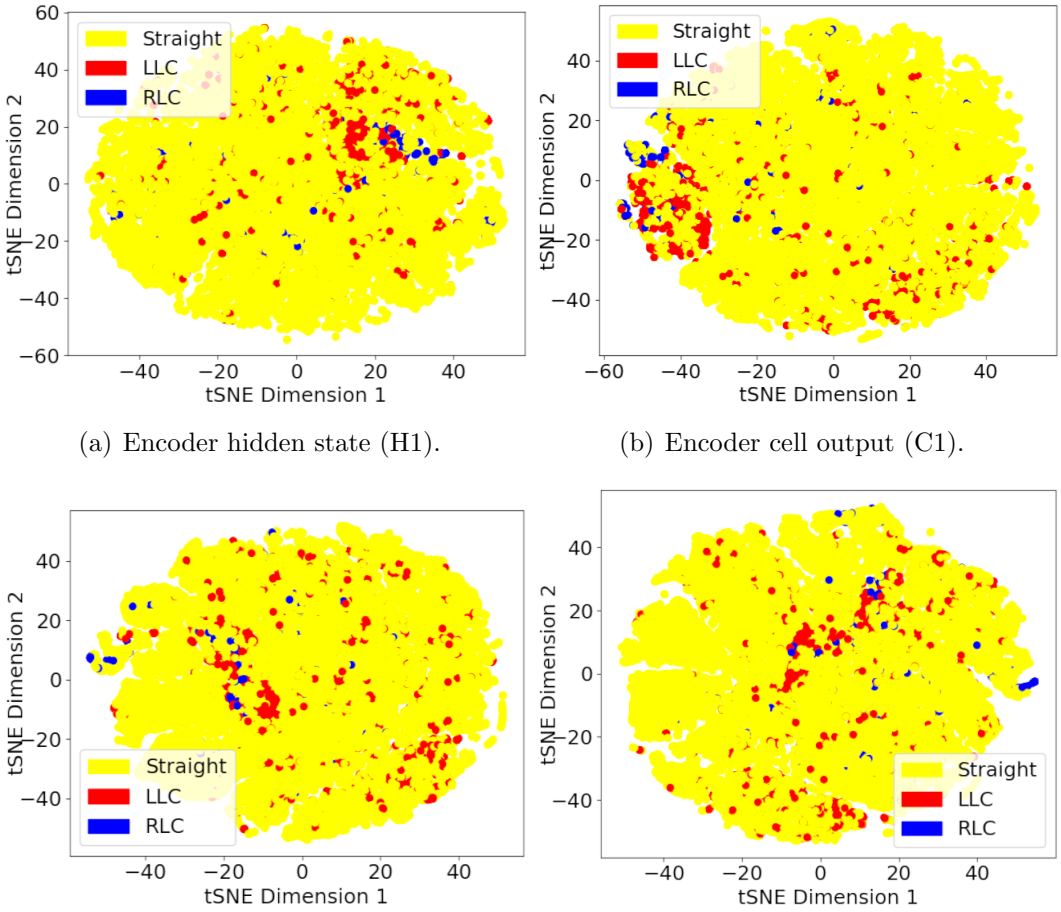
(c) Encoder hidden state (H2).

(d) Encoder cell output (C2).

Figure 4.11: Intermediate encoded state and cell output visualization using t-Distributed Stochastic Neighbor Embedding (t-SNE) [163] for the Lankershim intersection dataset.

Table 4.3: Accuracy and Precision for the three manoeuvre classes in all four datasets (Lankershim, Kingussie (T) junction, Edinburgh (Four-way) junction and I80/US101 highway) in comparison with the baseline method [123].

		manoeuvre Classes		
		Straight	**Left**	**Right**
Lankershim (Baseline [123])	**Accuracy**	0.858	0.925	0.933
	Precision	0.968	0.682	0.039
Lankershim (Ours)	**Accuracy**	0.958	0.964	0.972
	Precision	0.964	0.941	0.857
Kingussie (Ours)	**Accuracy**	0.999	–	0.999
	Precision	0.999	–	0.998
Edinburgh (Ours)	**Accuracy**	0.983	0.992	0.986
	Precision	0.977	0.986	0.982
I80/US101 (Ours)	**Accuracy**	0.972	0.991	0.975
	Precision	0.978	0.487	0.599



(a) Encoder hidden state (H1).

(b) Encoder cell output (C1).

(c) Encode hidden state (H2).

(d) Encoder cell output (C2).

Figure 4.12: Intermediate encoded state and cell output visualization using t-Distributed Stochastic Neighbor Embedding (t-SNE) [163] for both the I-80/US-101 highway dataset. Here LLC means left lane change and RLC right lane change.

Table 4.4: Class based normalized confusion matrix for the three manoeuvre classes at the Lankershim junction. Left, right and straight are left and right turn and straight on respectively.

		Actual		
		Straight	**Left**	**Right**
Predicted	**Straight**	0.963	0.038	0.088
	Left	0.023	0.941	0.056
	Right	0.014	0.021	0.856

Table 4.5: Class based normalized confusion matrix for three manoeuvre classes at Kingussie (T) junction (Radiate). Left, right and straight are left and right turn and straight on respectively.

		Actual		
		Straight	**Right**	**Left**
Predicted	**Straight**	0.999	0.002	–
	Right	0.001	0.998	–
	Left	–	–	–

Table 4.6: Class based normalized confusion matrix for three manoeuvre classes at the Edinburgh (Four-way) junction (Radiate). Left, right and straight are left and right turn and straight on respectively.

		Actual		
		Straight	**Left**	**Right**
Predicted	**Straight**	0.976	0.011	0.014
	Left	0.001	0.983	0.005
	Right	0.023	0.006	0.979

Table 4.7: Class based normalized confusion matrix for three manoeuvre classes in the I80/US101 dataset. Left, right and straight are left and right lane change and straight on respectively.

		Actual		
		Straight	**Left**	**Right**
Predicted	**Straight**	0.978	0.302	0.397
	Left	0.002	0.487	0.004
	Right	0.021	0.211	0.599

- **Constant Velocity (CV):** The instantaneous velocity of the TV was used to calculate the succeeding trajectory.

- **V-LSTM:** For the Vanilla-LSTM the encoder-decoder part of of the developed network architecture with the "trajectory-learning" section was only used. The output C_{t+1}^{D2} from the decoder was fed directly to "Trajectory-Learning" and not via "Manoeuvre" and "Velocity" learning. This made it possible to understand the performance improvement due to the proposed multi-stage prediction technique.

- **Dual-LSTM (D-LSTM):** D-LSTM is a two-stage LSTM network [176] where the first stage identifies the driver's intention, then the second stage estimates the future trajectory given the identified driver's intention.

- **Convolutional Social Pooling-LSTM (CS-LSTM):** This assigns a single LSTM cell to each vehicle, including the TV for the encoder. The decoder is only responsible for predicting the future trajectory recursively with an intermediate stage manoeuvre identification [29].

- **Multi-Agent Tensor Fusion (MATF):** Surrounding vehicle interactions and constraints were jointly modeled using a MATF technique [188] where past trajectories of the vehicles and scene contexts were encoded first, and then a convolutional fusion technique was adopted to capture multi-agent interaction.

- **SCALE-Net:** An edge-enhanced graph convolutional neural network was used [83] where each vehicle was considered as a node in the graph network and their interactions were captured by connected edges.

- **OGM Technique:** The occupancy grid map based approach proposed in the previous chapter. In this technique both the target and the surrounding vehicles were placed on a sequence of occupancy grid maps (OGMs) and then passed to a Conv-LSTM based architecture to capture the spatio-temporal features simultaneously. In addition to that an interactive prediction scheme was also developed. Before predicting the next time instance it uses the currently predicted positions of all the vehicles to update the OGM sequence, keeping the scene information up to date.

- **Proposed Ground-Truth (GT) Surrounding:** In general, during the training of an encoder-decoder architecture the decoder takes the GT information of the last time instance before predicting the current instance, but during testing it uses the predicted positions which causes a discrepancy. To identify the effect of this additional error the developed model was initially

tested with the predicted positions for the TV but the true positions for the SVs in the decoder.

- **Retrain Technique:** This is a more realistic setting where both the predicted positions for TVs as well as SVs were provided in the decoder. A method similar to "Schedule Sampling" [11] was adopted to train the model. Since the proposed approach selects the SVs dynamically, an offline training technique was adopted to eliminate the training-testing discrepancy. Instead of physically connecting the previous time instances of TV and SV's predicted positions to the current time instance in the decoder, the model was trained with the GT information in the decoder, i.e. "teacher-forcing". Once the model was partially trained the entire training set was predicted using the partially fitted model to repopulate the decoder inputs with the predicted future positions for the TV and dynamically selected SVs instead of using the GT information. Once populated the model was then retrained with this newly populated decoder inputs.

To perform a fair comparison the developed model was tested using 82000 test sequences randomly selected from both I-80 and US-101. Due to the absence of any intersections, the distance from the junction (j) and lane movement (μ) features were removed from the input feature set, with manoeuvre classes left lane change, right lane change and straight on. The normalized confusion matrix for these three classes and their accuracy/precision are in Tables 4.7 and 4.3 respectively. For D-LSTM, MATF and SCALE-Net, the results reported in their papers are considered, as the code or the pre-trained model is not available online.

A comparison of the RMSE between the predicted and ground-truth positions for a future horizon of 0.1 to 5 seconds is shown in Figure 4.13. The multi-stage trajectory prediction scheme (GT-surrounding) led to higher accuracies when compared to V-LSTM, as in the latter case the trajectory prediction is performed without the intermediate "manoeuvre" and "velocity" stages. Dual-LSTM achieved better performance than the naive CV model due to the intermediate manoeuvre recognition step. However, the CS-LSTM, MAFT, SCALE-Net, OGM, GT-surrounding and retrain techniques, all improved long-term prediction because Dual-LSTM lacks interaction in both the encoder and decoder. The additional road-map and traffic-rule features in this method (retrain technique) and the GT-surrounding technique achieved better accuracy for the entire prediction horizon compared to CS-LSTM , SCALE-Net and the OGM technique. In addition to this the feature vector based technique adopted in this chapter led to a significant reduction in the input data dimensions compared to the OGM technique (see section 4.4) but the accuracy of the predicted trajectories from both these methods are still comparable. In the case

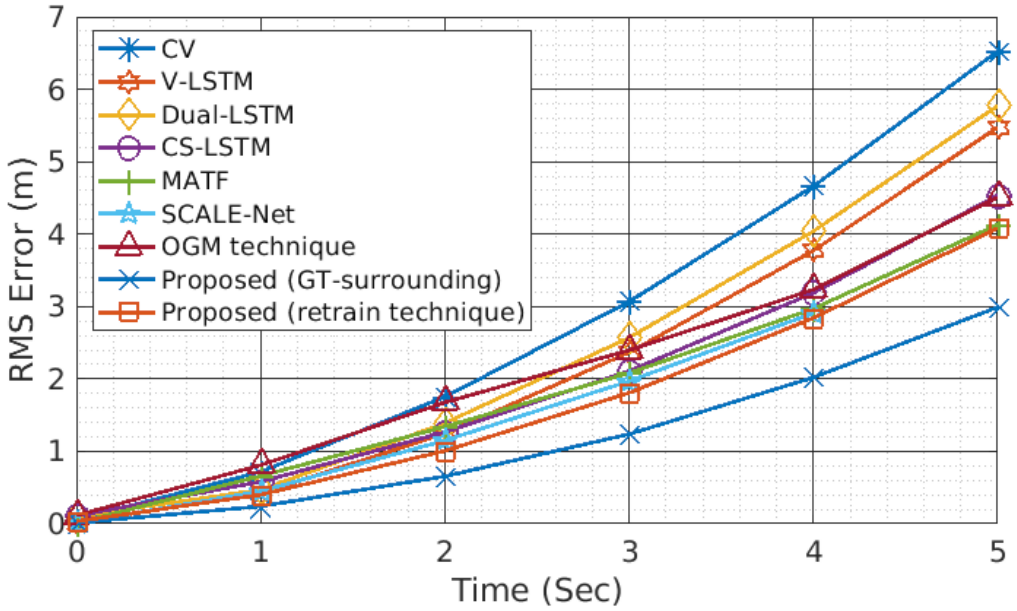


Figure 4.13: The RMSE comparison of CV, V-LSTM, Dual-LSTM [176], CS-LSTM [29], MATF [188], SCALE-Net [83], OGM, and the GT-surrounding and retrain techniques on over 82000 sequences randomly selected from the I-80 [24] and US-101 [23] datasets (NGSIM). The prediction time horizon is from 0.1s to 5s.

of MAFT, the scene context was embedded by an encoding channel but the missing interaction in the decoder led to a higher displacement error. To summarise, the method described in this chapter has achieved comparable accuracy with the state-of-the-art, but due to the classical encoder-decoder training-testing discrepancy, this has led to a higher displacement error compared to the GT-surrounding technique.

Assessment on the NGSIM Lankershim Dataset

To understand the impact of the additional intersection specific road map feature j and traffic rule feature μ the developed model was also tested on the Lankershim dataset. A comparison of the RMSE between the predicted and ground-truth positions for a future horizon of 0.1 to 5 seconds between the CV, V-LSTM, proposed GT-surrounding and retrain techniques, performed over 70000, randomly selected test sequences, is shown in Figure 4.14, and Table 4.9. Similar to the I-80/US-101 dataset the naive CV model produced the highest prediction error due to the missing temporal, interactive or map features. The GT-surrounding and retrain technique outperformed the V-LSTM technique due to the auxiliary "manoeuvre" plus "velocity" prediction stages, and the additional SV interaction in the encoder and the decoder. The classical training-testing discrepancy caused a higher displacement error for the retrain technique compared to the GT-surrounding technique.

Table 4.8: RMSE comparison on NGSIM I-80 [24] and US-101 [23] dataset between CV, V-LSTM, Dual-LSTM, CS-LSTM, MATF, SCALE-Net, OGM, GT-surrounding and retrain techniques at different time horizons.

Models	Prediction Horizon (meters) 1s	2s	3s	4s	5s
CV	0.71	1.75	3.06	4.66	6.52
V-LSTM	0.42	1.24	2.36	3.77	5.47
Dual-LSTM [176]	0.47	1.39	2.57	4.04	5.77
CS-LSTM [29]	0.58	1.26	2.11	3.18	4.53
MATF [188]	0.66	1.34	2.08	2.97	4.13
SCALE-Net [83]	0.45	1.15	1.97	2.91	–
OGM Technique	0.80	1.67	2.39	3.23	4.50
Proposed (GT-surrounding)	0.24	0.65	1.23	2.02	2.98
Proposed (retrain technique)	0.40	1.00	1.80	2.84	4.07

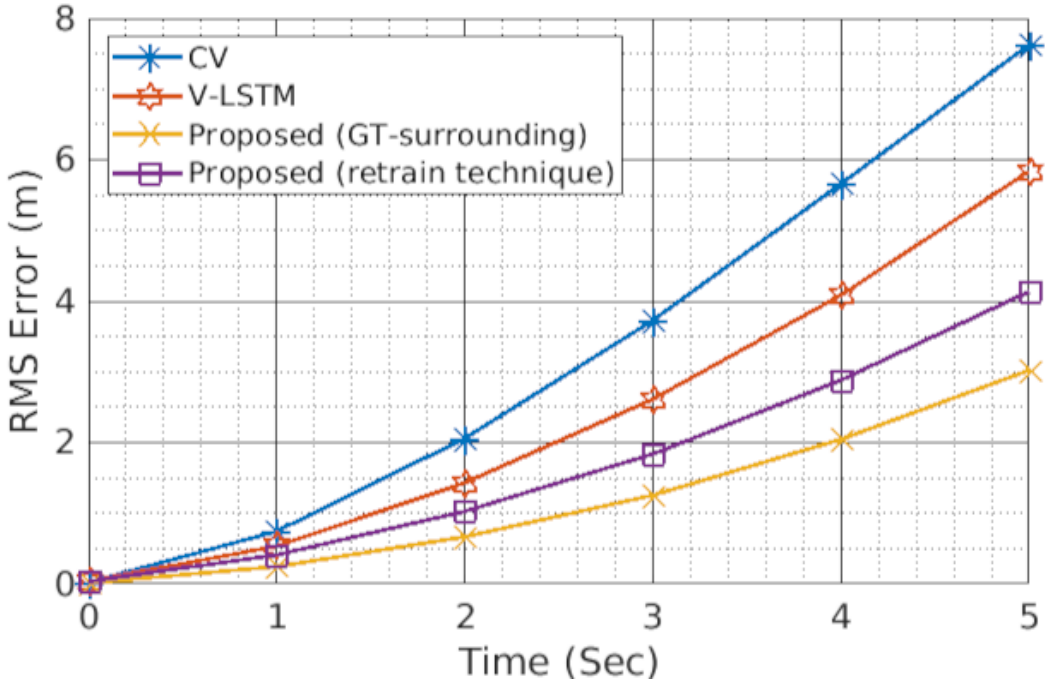


Figure 4.14: The RMSE comparison of CV, V-LSTM, proposed GT-surrounding and retrain techniques over 70000 sequences selected from the Lankershim Junction dataset (NGSIM) [25] dataset. This figure shows their average mean squared errors for the prediction time horizon from 0.1s to 5s.

Table 4.9: RMSE comparison on NGSIM Lankershim dataset [25] between CV, V-LSTM, proposed (GT-surrounding) and proposed (retrain techniques) at different time horizons.

Models	Prediction Horizon (meters) 1s	2s	3s	4s	5s
CV	0.74	2.05	3.72	5.65	7.62
V-LSTM	0.53	1.42	2.61	4.08	5.83
Proposed (GT-surrounding)	0.24	0.66	1.25	2.04	3.02
Proposed (retrain technique)	0.41	1.02	1.83	2.88	4.13

Assessment on the "Radiate" Dataset

Additional evaluations with the newly collected radar dataset were performed. The motivation is to see whether the proposed approach is effective within the field of view of an automotive radar, which is the main concern. The two datasets used were those collected at the Kingussie (T-junction) and Edinburgh (Four-way) junctions. The RMSEs on these two junctions are shown in Figure 4.15. and tabulated in Table 4.10.

As the developed architecture is predicting the future motion ("Motion Learning") as an intermediate stage, its performance was also evaluated by recursively calculating the current position using the last calculated position at the previous time instance and the current predicted motion for the entire future horizon (1s to 5s):

$$\hat{x}_{t+1} = \hat{x}_t + \hat{\dot{x}}_{t+1}, \quad \hat{y}_{t+1} = \hat{y}_t + \hat{\dot{y}}_{t+1} \tag{4.9}$$

where $(\hat{x}, \hat{y})_{t+1}$ and $(\hat{x}, \hat{y})_t$ are the calculated positions for the next and current time instances respectively, and $(\hat{\dot{x}}, \hat{\dot{y}})_{t+1}$ is the predicted motion for the next time instance. In Figure 4.15, the RMSEs of the trajectories generated directly through "Trajectory Learning" and 'Motion Learning" are denoted as $junction_{pos}$ and $junction_v$ receptively $\forall junction \in [Kingussie, Edinburgh]$.

As stated earlier, the radar system we have used has only a 4 Hz frame rate whereas the NGSIM dataset has a 10 Hz frame rate. Despite the 2.5 times lower frame rate, the developed architecture achieved almost similar performance. The longer trajectory sequences recorded with cameras installed on top of high buildings (NGSIM) helped the model to understand different driving behaviours as opposed to the relatively shorter trajectory sequences collected with the more limited range radar, making the problem even more challenging. Due to the simpler junction

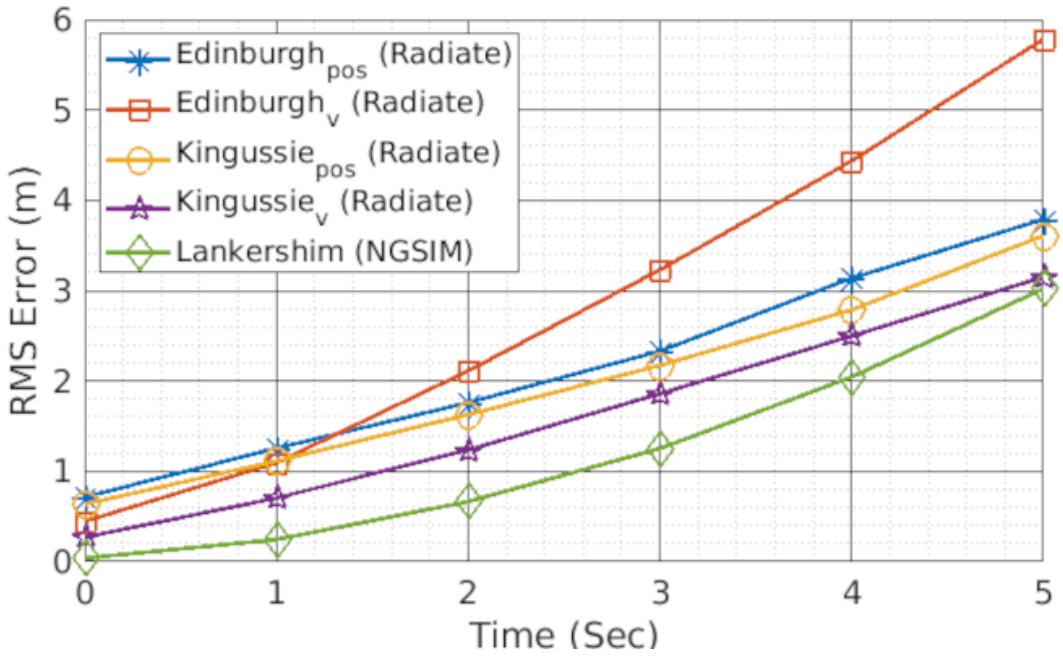


Figure 4.15: The RMSE comparison of Lankershim (NGSIM), Kingussie (T) and Edinburgh (Four-way) (Radiate) datasets. This figure shows their average mean squared errors for the prediction time horizon from 0.25s to 5s.

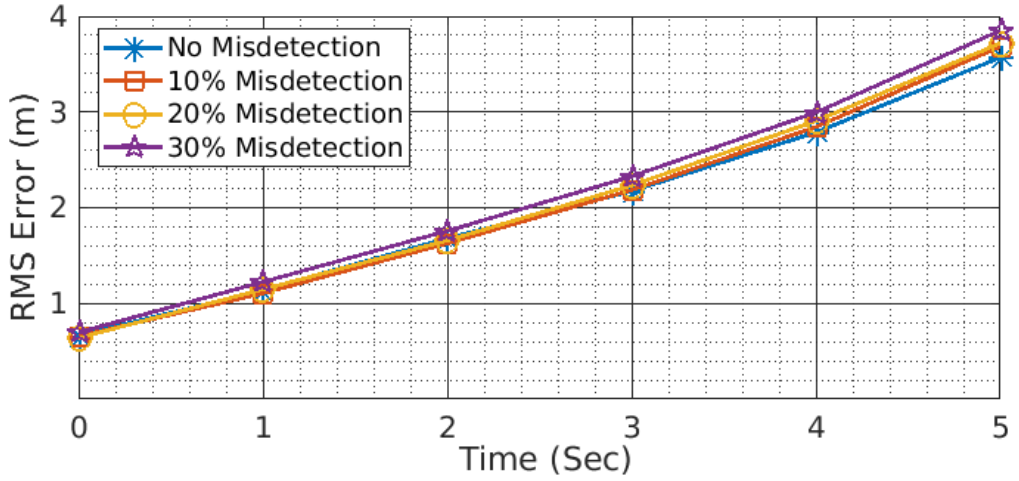
Table 4.10: RMSE comparison on the Radiate dataset between the Edinburgh (Four-way) and Kingussie (T) junctions at different time horizons.

		Prediction Horizon (meters)					
Junction	**Technique**	**0.25s**	**1s**	**2s**	**3s**	**4s**	**5s**
Kingussie	**Motion**	0.27	0.70	1.23	1.85	2.49	3.15
	Position	0.63	1.11	1.62	2.17	2.78	3.60
Edinburgh	**Motion**	0.44	1.09	2.11	3.23	4.43	5.83
	Position	0.71	1.25	1.76	2.33	3.13	3.78

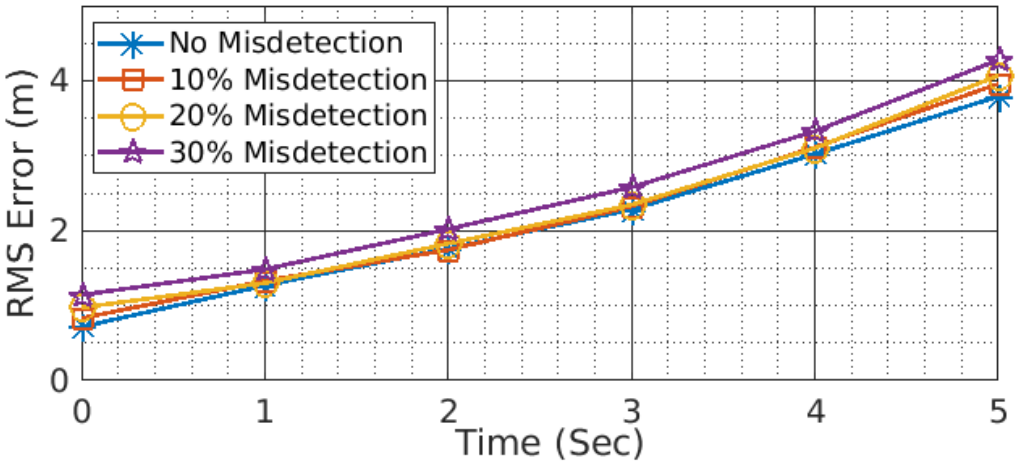
structure at Kingussie, the RMSE was lower than the Edinburgh junction. The RMSEs using "Velocity Learning" were lower in the short-term horizon compared to those predicted directly through "Trajectory Learning" (see Table 4.10 and Figure 4.15), because in the short term the current velocity has a more prominent effect on future position, but in the longer term, the surrounding vehicle positions and scene context have a beneficial effect. A demo of this work can be found here.

4.7.3 Reliability with missed detections

The evaluation to date is on perfectly annotated vehicle positions (as with NGSIM), but for a live system this is unrealistic as there will always be missed detections. To understand better the impact of this on performance additional experiments with 10,



(a) Kingussie (T) junction



(b) Edinburgh (Four-way) junction.

Figure 4.16: The RMSE comparison on the Radiate dataset [143] with different mis-detection rates, for the prediction time horizon from 1 to 5s.

Table 4.11: RMSE comparison on the Radiate dataset for 10, 20 and 30% mis-detection rates at different time horizons.

		Prediction Horizon (meters)				
Junction	**mis-detection**	**1s**	**2s**	**3s**	**4s**	**5s**
Kingussie	**10%**	1.10	1.62	2.18	2.84	3.68
	20%	1.14	1.65	2.23	2.90	3.71
	30%	1.22	1.75	2.32	2.99	3.84
Edinburgh	**10%**	1.32	1.74	2.32	3.11	3.95
	20%	1.30	1.82	2.34	3.10	4.06
	30%	1.48	2.02	2.57	3.32	4.27

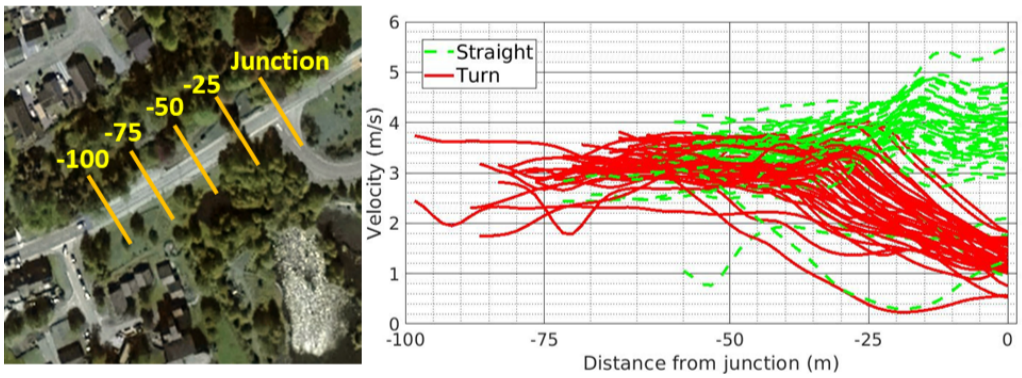
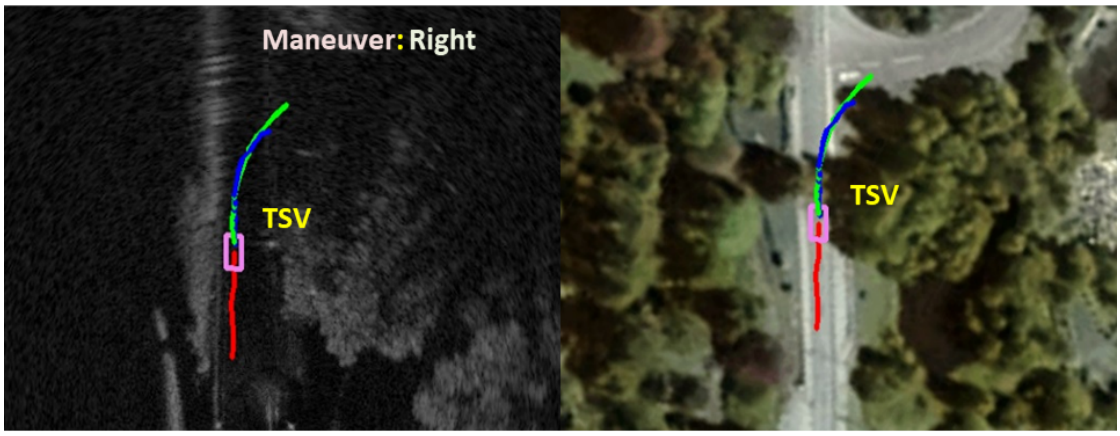


Figure 4.17: Detected vehicles' velocity profiles from 100 metres till the junction location at Kingussie (T-junction). Yellow lines in the satellite image [110] show the distance from the junction in metres. Solid red and dashed green lines in the plot show the velocity profiles of the turning and straight-on vehicles, respectively, at different distances from the junction.

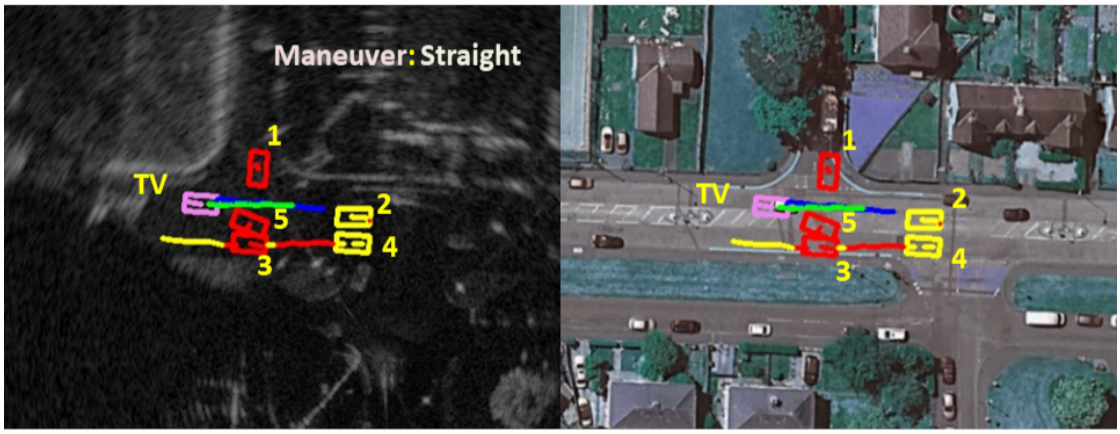
20 and 30% missed detections were also performed. During training the full, correct data was considered, but during testing simulated mis-detections or occlusions were created by intentionally dropping the TV and SV positions for consecutive frames in the input sequence. Separate particle filters were maintained for each tracked TV and SV and the the two conventional particle filtering stages, i.e. particle propagation and weight update on their assigned particle filters, were applied to re-estimate those dropped TV and SV positions. At first the current particles were propagated using the last known velocity. Since there are no new observations, instead of updating the weights of the particle filters, equal weights were assigned to all which were then used to estimate the dropped positions before passing to the network. The RMSE comparisons between perfect and missed detection data for the Kingussie (T-junction) and Edinburgh (Four-way) junctions are shown in Figs. 4.16(a) and 4.16(b) respectively and reported in Table 4.11. In both cases it achieved only slightly poorer performance which shows that the proposed technique is robust to imperfect detection or potential occlusion.

4.7.4 Qualitative analysis and case studies

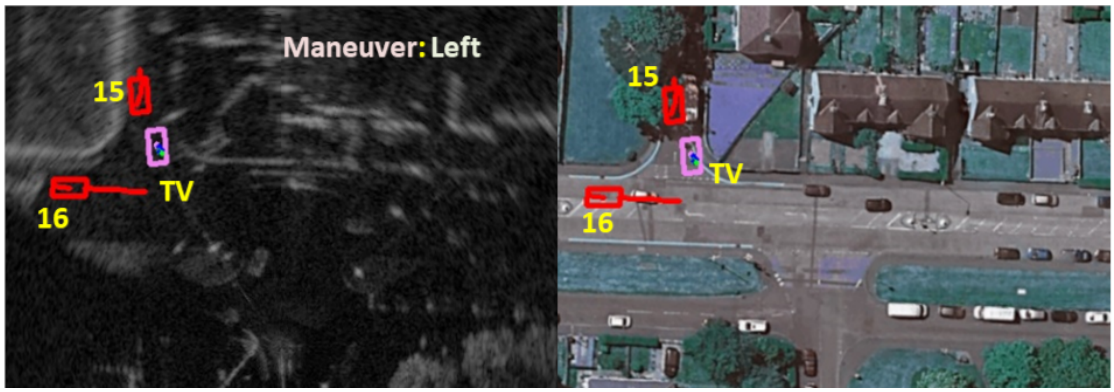
To better illustrate the proposed approach, four different examples are shown in Figs. 4.18(a) to 4.19(c). On the top left, a simple turn is shown. There are no SVs, but analysing the velocity profiles of the straight and turning vehicles in Figure 4.17, it is clear that the vast majority of turning vehicles have a decelerating velocity profile when compared to straight vehicles moving at a constant speed. It is likely that another feature, like distance from the junction, plays a significant role. The four-way junction at Edinburgh has considerably more vehicle interaction and is shown



(a) Successful right turn prediction at Kingussie (T) Junction [110]

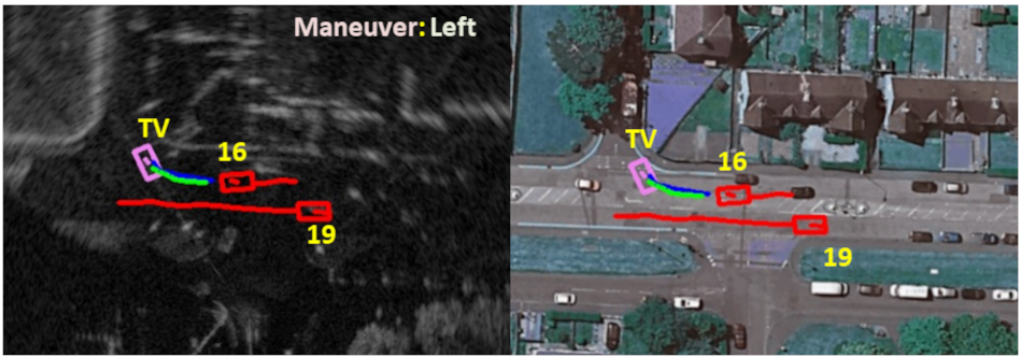


(b) Right-of-way for the vehicles travelling on the main road [108].

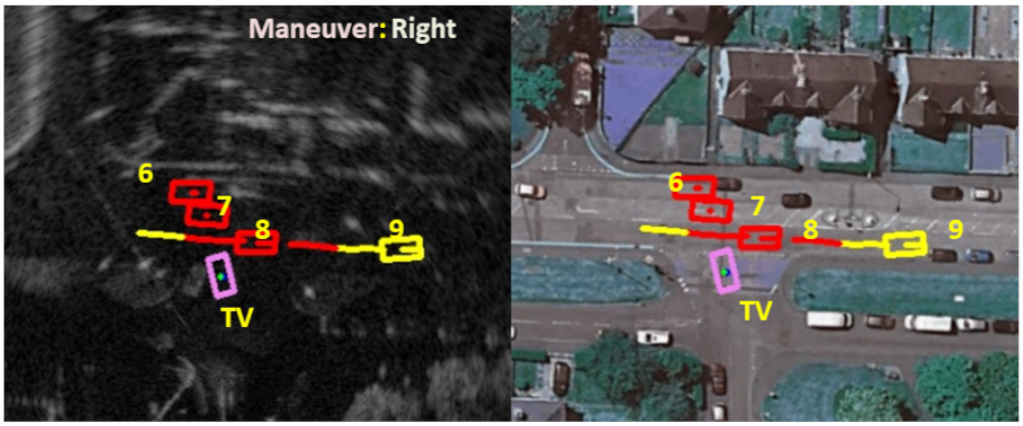


(c) Vehicles on the side road [108] cannot merge onto an occupied main road.

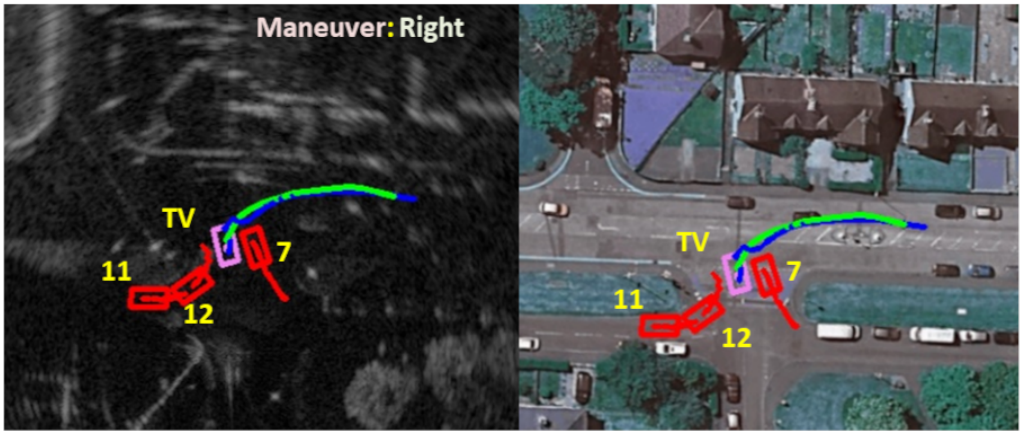
Figure 4.18: Qualitative analysis of the prediction results of the proposed technique. For each subfigure, the left and right are views from the radar image and satellite [110], [108], respectively. The violet car marked as TV is the Target Vehicle (for which the prediction is being performed). Predicted and ground truth trajectories for the TV are drawn in blue and green lines, respectively. Red vehicle and trajectories indicate the currently selected SVs by the decoder as opposed to yellow vehicles and trajectories that indicate un-selected vehicles (not SVs) by the decoder based on the Euclidean distance from the TV. For each vehicle, the future trajectory was plotted for 5 seconds.



(a) Vehicles on side road [108] can only merge onto main road when its empty.



(b) Selection of effective surrounding vehicles in the decoder based on the current scene [108]



(c) Merging on main road [108] after it is clear

Figure 4.18: Qualitative analysis of the predictions of the proposed technique. For each subfigure, the left and right are views from the radar image and satellite [110], [108], respectively. The violet car marked as TV is the Target Vehicle (for which the prediction is being performed). Predicted and ground truth trajectories for the TV are drawn in blue and green lines, respectively. The red vehicle and trajectories indicate the currently selected SVs by the decoder as opposed to the yellow vehicles and trajectories that indicate un-selected vehicles (not SVs) by the decoder based on the Euclidean distance from the TV. For each vehicle, For each vehicle, the future trajectory was plotted for 5 seconds.

in Figure 4.18(b). The TV is on the main road with SV-1 waiting on a side road to merge and SV-5 waiting on the main road to turn into the same side road. Despite multiple surrounding vehicles, with knowledge of its own lane location and right-of-way (the vehicle on the main road has priority), the proposed technique predicted a straight, constant speed manoeuvre as opposed to any sudden stop. The opposite effect of the same right-of-way rule is shown in Figure 4.18(c), where the TV is on the side road and the main road is occupied by SV-16. Due to congestion on the main road the proposed model predicted a stop manoeuvre and a wait till the main road is clear. As shown in Figure 4.19(a) once SV-16 crosses the side road with no SVs on its tail, the main road is empty and this allows the target vehicle to merge safely.

Figure 4.19(b) and Figure 4.19(c) demonstrate a potential fatal consequence that can happen by assigning specific surrounding vehicles at the beginning of the observation sequence. As shown in Figure 4.19(b), the model assigned vehicles 6, 7 and 8 as the considered SVs on the basis of Euclidean distance at the beginning of the prediction sequence. Keeping the current situation in mind any real driver would consider these same vehicles before merging in the main road. However, once SV-8 crosses the side road at a safe distance, it becomes irrelevant after a few predicted frames. On the other hand, a different SV-9, not close enough to be considered initially, becomes relevant. A human driver should now consider SV-9 and ignore SV-8 before deciding to merge onto the main road. An approach that assigns specific LSTM cells to the current surrounding vehicles at the beginning of the sequence [74], [29] does not allow reconsideration of incoming vehicles, and could lead to a dangerous manoeuvre.

4.8 Summary

A novel encoder-decoder architecture for vehicle behaviour prediction was presented in this chapter. A key contribution is the encoding of the surrounding road network and vehicle interaction, not just in the encoder but also in the decoder to keep the scene information up to date. A further innovation is the consideration of the relevant, surrounding vehicles, updating the network on the basis of the nearest rather than a fixed set of nearby vehicles. This allows the birth and death of vehicles as they come into and disappear from view. Using the encoder state with the estimated manoeuvre and motion intention, this makes a well informed future estimation of manoeuvre and trajectory. Several experiments on a previous, public dataset (NGSIM) show that the proposed technique can achieve comparable prediction accuracy with the state-of-the-art. Evaluating this same architecture on new data collected from an automotive radar with very different sensing characteristics

shows very encouraging performance in spite of the much more constrained field, slower frame rate and shorter range of view.

Chapter 5

Conclusions and Future Directions

This work has focused on the accurate anticipation of the future maneuvers and trajectories of road vehicles in the immediate vicinity of an assumed ego vehicle as an aid to autonomous or assisted driving. Two distinct architectural conributions have been made to the development of LSTM architectures for behaviour prediction based on sensor data. The methods developed have been evaluated under various traffic conditions including a multi lane motorway, a single lane, un-signalled T-junction, a signalled multi lane four-way intersection, and finally an un-signalled single lane four-way intersection.

Data Collection:

The team equipped a van with a hardware sensor suite that includes a 77-76 GHz automotive Navtech radar, a 32 channel Velodyne liDAR, a ZED stereo camera and a Advance Navigation GPS/IMU tool kit. The software framework, data acquisition, processing and storage into a meaningful structure was developed using the Robot Operating System (ROS) and MATLAB. Although there are many public datasets available, the majority do not include radar data, and also the data is distinct in including labelled radar images. The data was collected in both static and moving ego-vehicle settings at the various traffic locations.

LSTM architecture based on OGMs:

A new Conv-LSTM architecture was described in Chapter 3, showing the benefits of extracting spatio-temporal features simultaneously. This used a sequence of occupancy grid maps (OGMs) as input to predict the future vehicle trajectories. An interactive feedback based prediction scheme was developed to predict all the vehicle positions for the future time instances. When predicting the next instances the predicted positions for the current instance were combined with the input OGM sequences. This interactive predicting scheme has the capability to update the scene information for a long term prediction horizon, thus improving the accuracy of the predicted trajectories. The results were compared with state-of-the-art trajectory

prediction techniques using two publicly available bird's-eye view datasets, NGSIM [23], [24] and High-D [89]. Moreover, to establish the generalised nature of the proposed architecture the model was trained and evaluated on different highway scenarios, i.e. in the case of NGSIM the model was trained on the US-101 [23] dataset and tested on the I-80 [24] dataset.

An improved encoder-decoder LSTM architecture:

Dispensing with the OGM model, An improved, novel encoder-decoder architecture was described in Chapter 4. The context information, i.e. the information about the surrounding vehicles, was fed not only into the encoder but also into the decoder to update the scene data during long term prediction. In addition, a dynamic surrounding vehicle selection scheme was developed, identifying and selecting the most influential surrounding vehicles using the Euclidean distance between the target vehicle and all the other vehicles present in the scene. This is important because, instead of fixing the surrounding vehicles at the beginning on the observation sequence as is commonly done, this dynamic selection scheme allows the model to accept the birth and death of vehicles as they appear and disappear in the target vehicle's vicinity. As an additional contribution, rather than predicting the future trajectory directly through the decoder, an auxiliary architecture was developed to predict the future maneuver first, followed by the future motion and finally the future trajectory.

The model described in Chapter 4 was evaluated on both publicly available multi-lane highway [24], [23] and multi-lane intersection [25] data to compare the performance with the state-of-the-art, achieving comparable prediction accuracy. The model was also tested on the newly collected "Radiate" dataset. In both the cases the proposed model achieved comparable performance to the publicly available BEV data despite the fact that this data has a more constrained field-of-view, a lower frame rate and limited range. Comparing the Kingussie (T) junction and the Edinburgh (four-way) junction, the Kingussie (T) junction yielded a higher prediction accuracy due to the simplicity of the junction structure and the clear contrast between the turning and straight-on vehicle's velocity profiles.

An additional evaluation, contrasting with the perfectly human annotated data used so far, included artificial mis-detections that were created by dropping consecutive frames in the "Radiate" dataset. This simulated the effect of occlusion or detection errors, when for example the cars are detected and classified through a deep learning based network. The near-similar performance in both the perfect and imperfect data shows that the model has some degree of robustness.

Summary: Despite significant progress in vehicle behaviour prediction, one key, missing factor has been consideration of the latest predicted behaviours of all

the surrounding vehicles while performing prediction for the next instance. This missing stage has made it difficult in previous models to keep the scene information up to date during the long term,, leading to poorer predictions. In contrast, both the developed models in this thesis considered future predicted positions, and sometimes the predicted manoeuvres and velocities as well, before predicting further ahead. This has led to an improved accuracy of the predicted trajectories from the methods described in Chapter 3 and 4. In addition to this the model in chapter 4 identifies and includes the most important surrounding vehicles dynamically at each timestamp for the entire observation and prediction horizon, instead of fixing them at the start, thus making it more relevant in any continuously evolving traffic scene.

5.1 Limitations of Proposed Approaches

Although the models presented in this thesis show promising results in various traffic conditions, there are a number of significant limitations:

- **Intersection Specific:** In the case of the "Radiate" dataset the same architecture was used for both the Kingussie (T) and Edinburgh (four-way) junctions but it needed retraining. In other words the model was retrained with the respective datasets before testing, instead of using a single set of model weights produced by training the model with a mixture of vehicle trajectories selected from both datasets. This means that the model was not general, but junction-specific.

- **Missing Road Layouts:** Although the models were evaluated on both left-hand and right-hand driving systems as well as at various traffic conditions i.e. highways and intersections it was not tested exhaustively on other possible layouts, e.g. roundabouts that are common in urban locations. In the case of Chapter 4 both the target and surrounding vehicle positions were passed as a feature vector format without maintaining any fixed orientation. This suggest that the proposed technique can be extended to other scenarios (e.g. a roundabout) but only after adding some additional priority rules either to the model or through features encoded in the input feature vector. However, for the OGM technique proposed in chapter 3 the orientation of each OGM is fixed for the entire observation and prediction sequence. In this case, near any intersection or roundabout, the OGM sequence will be created considering the main road as the initial alignment. Using the fixed width of each OGM the model will be able to predict the future trajectory of all the turning vehicles up to a point where the vehicles leave the main road in order to enter in any of the side roads. Once the turn maneuver is complete a new set of OGM sequences

should be created but this time aligned with the side road the target vehicle entered.

- **Limited Road Actors:** In the current context only vehicles were considered. Other important road actors such as pedestrians, bicycles, motorbikes etc were not included. In the case of highways this may not cause much impact, but they are common in any urban scenario. Adding these different types of road actors into a single model and capturing their interactions is not straightforward as they all behave differently. For example, a vehicle can never perform a sudden 180 degree turn at its current location because of the constrained front wheel turn angle whereas that is not the case for a pedestrian. Moreover the range of speeds at which individual road actors can move are completely different. A vehicle can move at any speed from 0 to 100+ mph depending on circumstances, Whereas the average speeds of pedestrians and cyclists are somewhere in between 3-4 mph and 10-20 mph, respectively.

5.2 Future Work

The above mentioned limitations should be addressed with the following additional steps:

- **More Intersection Data:** Till now a very limited amount of trajectory data was collected using the test vehicle. This made it harder to develop a generic model which will work at any intersection without retraining. Additional data at a variety of diverse intersections would be beneficial, such as vehicle trajectory data collected from multiple T-junctions, four-way intersections, 3, 4 and 5 exit roundabouts etc. Then a single model could be trained with a much greater variety of vehicle trajectory data to make it more generic, and applicable unseen intersections. In addition it would be helpful to extract the type of intersection using the Google satellite map and use this during both the training and prediction process. This would help the model to segregate different driving behaviours at various types of intersections and then use that prior information while predicting the future trajectory at any type of intersection.

- **Additional Road Users:** Attributes associated with other road actors (pedestrians, bicycles, motorbikes) should be added to the vehicle data and features to make better informed future trajectory prediction. Different road actors have different attributes. For a vehicle the important attributes are its current location, velocity, and acceleration, coupled with various map based features

such as lane number, movement rules associated with that lane, distance from the nearest junction etc. However, for a pedestrian these will change to how far he/she is from the end of curb, gaze direction, whether travelling solo or in a group, wearing any headphones or using his/her mobile etc. All these features can help a model to learn how a pedestrian behaves. To develop a model which can capture interaction between different road actors interaction, the input feature vector must be populated accordingly before passing it to the model both during training and prediction. This requires better perception algorithms that not only identify the location, motion and type of actor (pedestrian car etc.), but also details about their current state(using phone, gaze direction etc.)

- **Adverse Weather and Light Conditions:** The surrounding vehicle information was extracted using the Navtech Radar. Although the radar signals have been shown to be relatively unaffected by adverse lighting or weather conditions, it is still important to evaluate the model with adverse weather data, e.g. a rainy or foggy day or at the night time. To do that at first the relevant data sequences must be selected from the "Radiate" dataset. Once done all the vehicles from those sequences would be annotated to generate the ground truth vehicle trajectory data. This step can be done either by human annotation directly or conceivably using deep learning based pre-trained vehicle detection with error correction. This annotated, poor weather vehicle data would be used to evaluate and compare the developed behaviour prediction model performance against the data collected during good weather and daylight conditions.

- **Detection-Tracking-Prediction:** Although an evaluation with missed detections has been included in this work, the crucial step would be to develop an end to end processing system for automotive radar, marrying vehicle detection, and multiple object tracking to behaviour prediction. In the first stage, a state-of-the-art deep learning based vehicle detection model can be used to detect all the vehicles from the automotive radar images in the form of bounding boxes. Once detected, these are passed to a tracking filter such as the GM-PHD filter to remove noise and clutter, then data association to match vehicles across frames, generating continuous trajectories. The used tracker should also be capable of doing "birth" and "death" of new and existing vehicles, respectively. These continuous trajectories with their associated map features would be passed to the pre-trained vehicle behaviour prediction model to anticipate how the scene will evolve in the near future.

- **Situation Based Model Selection:** As mentioned earlier, depending on the current traffic location of the target vehicle the input feature vector set needed for future trajectory prediction changes. Instead of a completely general, unlikely model, a multi-model approach can be developed where individual models are responsible for different traffic situations, e.g. multi-lane highway, intersection or roundabout. This could, for example be selected from a GPS and map data. In other words the autonomous ego vehicle's current location will be extracted from its on-board GPS sensor which can access a detailed digital map to identify the type of traffic location in which it is currently travelling. Once identified, the relevant pre-trained model will be selected and used to predict the future trajectory of all the other vehicles in the vicinity.

Appendix A

Test Vehicle and Algorithms

This appendix mainly covers two major areas.

- The "Radiate" dataset used in this thesis is taken from an 'ego-vehicle'. We consider here the drawbacks of the existing datasets as well as the main motivation behind creating this new dataset. The hardware used to setup the test vehicle and the developed software framework used to collect and store the raw data are described. A list of traffic locations where the data was collected is given.

- Pseudo-codes are given of the developed algorithms performing the interactive vehicle trajectory prediction tasks both in chapter 3: Interactive Vehicle Trajectory Prediction with Convolutional Recurrent Neural Networks and chapter 4: Predicting Vehicle Behaviour using Automotive Radar and Recurrent Neural Networks.

A.1 The "Radiate" Dataset

Collecting real world traffic data to validate the developed trajectory prediction algorithms is critical. Proper measures must be taken while collecting the data to ensure closeness to the real, ego-vehicle perspective. This is important because, if the developed algorithms need to be used with vehicle on-board sensors, there is no guarantee that they will perform similarly to the data collected with the overhead sensors using a bird's-eye view. Even though in the near future infrastructure sensors will play a crucial role to avoid any potential collisions, e.g. at blind intersections, there will still be a significant overlap with the ego vehicle on-board sensors. Moreover, there will be many intersections without the presence of any infrastructure cameras. Preferably, the collected data must be 100% natural, which means the driver driving the sensor equipped test vehicle must be allowed to drive normally

and all road actors in the vicinity should not be aware that their movements are being collected. Only then can the captured dataset include various driving behaviour anomalies such as over speeding, careless driving, aggressive overtakes on motorways or negligent merging onto the main road at intersections etc.

Considering the challenges of a natural vehicle trajectory dataset a lot researchers used off the shelf simulators such as IPG CarMaker, PTV Vissim [182] or Carla [37] etc. to generate the dataset which could then be used to train and validate the developed algorithms. But the major discrepancies between a dataset collected with on-board sensors and through simulators are as follows:

- **Data Inaccuracies:** Any simulator-generated synthetic data assumes perfect knowledge of the scene which is not true in the real world. For example the localization of the ego-vehicle is usually done through on-board GPS systems which will not always be accurate due to the canyon effect [139], especially when it is travelling through an urban area surrounded by high rise buildings. Usually all the surrounding vehicle positions are estimated using their relative position with respect to the ego-vehicle location. Now, any error in the ego-vehicle location is also accumulated in the surrounding vehicle location.

- **Occlusion Free:** The assumption of perfect perspective of all the other road actors in the scene can make the simulator data very dissimilar to the real world data. Using vehicle on-board sensors can often lead to a situation where the critical and small road actors such as pedestrians, bicyclists, or motorbikes will be occluded by the other big road actors such as vehicles, trucks or lorries. Introducing such natural occlusions in any simulator can be very laborious.

- **Perfect Detection and Tracking:** All simulators assume an accurate detection of all the other vehicles in the vicinity, then noise free relative location estimation with respect to the ego-vehicle, and finally perfect association of all the other targets across frames to generate the continuous trajectories. This is again not realistic as missed detection and wrong association of two different targets across frames due to occlusion by other road actors are very common phenomena in congested traffic.

- **Perfect Driving:** It is very difficult to make a simulator drive a vehicle exactly as a human does. The driving behaviour in any simulator is always perfect which means it does not include any extreme cases and driving irregularities such as close cut-ins or rash driving which are very common in real traffic.

Considering the above mentioned limitations in any simulator, there is no guarantee that if a developed algorithm works well with synthetic data, it will have

similar performance on real data. This was the main motivation behind collecting and publishing an ego-vehicle perspective multi-sensor dataset called "Radiate" [143]. The data collection was done using a sensor equipped Volkswagen Transporter while driving around different places in Scotland, United Kingdom. The collected data consists of camera images, laser scans, high gigahertz radar images, high precision GPS and IMU measurements from a combined GPS/IMU kit, shown in Figure. A.1. The main purpose in the context of this thesis in setting up the test vehicle with different sensor modules and collecting real-time driving data is to push forward the development of trajectory prediction algorithms targeted to different traffic scenarios (motorways, junctions, round-abouts etc). The usage of radar made the dataset useful even in adverse weather and light conditions (rain, fog, night etc). The collected traffic data was also used by our research group for various additional tasks in the self-driving area, like visual-odometry/Simultaneous Localization and Mapping (SLAM) [72], vehicle detection/tracking in radar images [144], and radar/LiDAR fusion for scene mapping [167]. After presenting an in-depth review of existing ego-vehicle perspective datasets this appendix provides details about the sensor equipped test vehicle and the Robot Operating System (ROS) based software framework that was developed to capture and store the raw sensor data. Lastly, it describes the various traffic locations used in this thesis, where the test vehicle was either parked or driven through, to collect diverse traffic data.

A.1.1 Existing Ego-vehicle Perspective Real Datasets

To make the benchmark datasets more realistic there has been a recent increase in the collection and publication of surrounding scene data from the ego-vehicle point of view, not with overhead sensors, for which various sensor equipped test vehicles have been driven for many hours at different locations. One commonly used dataset is the KITTI Vision Benchmark suite [53] which includes four different sensor modalities i.e. stereo camera, 64 beam Velodyne LiDAR, Global Positioning System (GPS) and an Inertial Measurement Unit (IMU), collected in different European cities. Similar datasets collected with sensor equipped ego-vehicles include comma.ai's dataset [130], the Udacity dataset [159] and Brain4Car's dataset [80], mostly collected in the San Francisco bay area. Other large datasets collected in different cities of the USA are CityScapes [27], ApolloScape [168] and UC Berkeley BDD100k [185], but they all consist of only camera and depth information using stereo pairs. The more recent, Oxford RobotCar Dataset includes all the above mentioned sensor modalities [103] along with a high resolution radar [10] but the dataset is mainly focused on radar odometry, and is collected in urban areas (Oxford city). In order to collect the data, the test vehicle travelled over the same route mul-

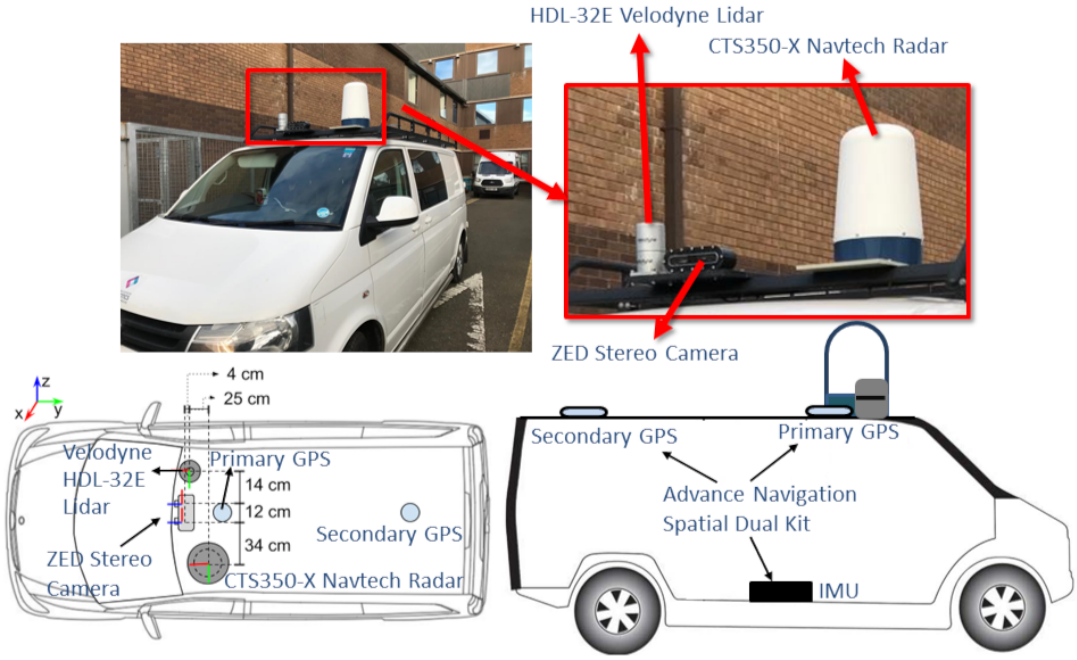


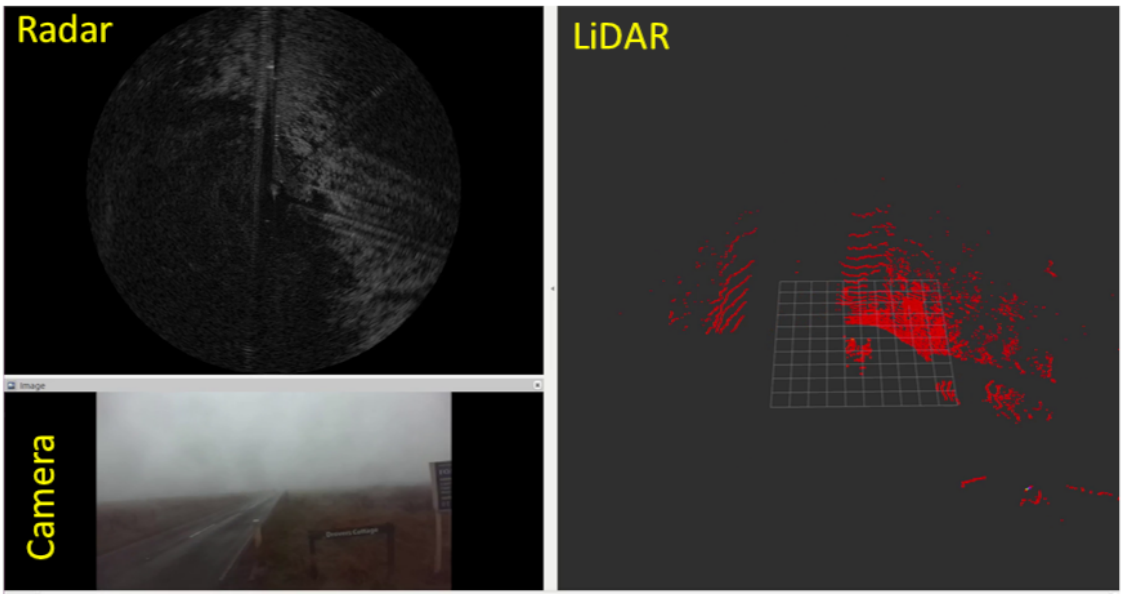
Figure A.1: Schematic and real views of our test vehicle used to collect data. The top left image is the Volkswagen Transporter van equipped with LiDAR, radar, stereo camera and GPS/IMU kit. On the top right is a close up view of the Velodyne LiDAR, CTS350-X Navtech Radar and ZED Stereo Camera. On the bottom left is the schematic top view showing the individual sensor positions except for the IMU as it is placed on the floor inside the vehicle. On the bottom right is the schematic side view showing the exact IMU and GPS placements.

tiple times to create radar based ground truth odometry information. Moreover, the dataset contains only raw radar images without any surrounding vehicle annotations or continuous trajectory information, which makes it hard to use in any trajectory prediction, or indeed vehicle detection or classification, algorithm. Another similar type of dataset, "nuScenes" [16], also includes radar data points collected with six, 77GHz Frequency Modulated Continuous Wave (FMCW) radars along with RGB cameras, one LiDAR and one GPS/IMU tool kit. However, it only provides a sparse 2D point cloud instead of the full radar waveform, which again makes it very hard to perform object detection and/or classification tasks. There are even cases where the on-board radar returned only a single point for an entire vehicle in its range. Clearly it isn't possible to perform detection and classification of different road actors using a single radar point return.

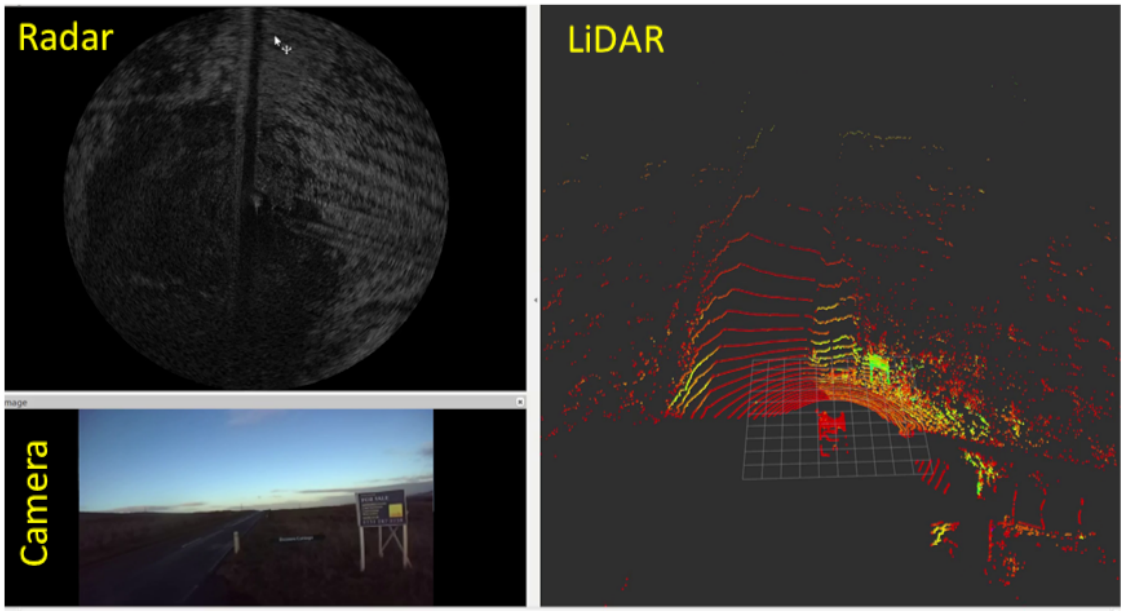
Although all these datasets collectively cover a wide variety of traffic scenarios and sensor modalities, they were collected mostly in ideal weather/light conditions and the test vehicles were always in continuous motion, which means none of them stayed in any specific location or intersection for a longer period of time to collect different intersection specific traffic flow. Keeping these two limitations in mind we set up our test vehicle with different sensor modalities including high resolution radar and collected our own dataset at different locations in Scotland, UK. We focused mainly on signal-free intersections and adverse weather conditions. The signal-free intersections helped us to understand the negotiations that usually happen between vehicles without any interventions from the traffic infrastructure, such as multi-lane roads with specific turn rules or traffic lights. These traffic infrastructures often play a significant role in ordering the vehicle flow at any dense intersection. The adverse weather conditions helped us to understand the impact of weather and light conditions on different sensor modalities. A typical comparison of different sensor data collected at the same location in different weather conditions, i.e. with and without fog, are shown in Figure. A.2(a) and Figure. A.2(b), respectively, where both the Velodyne LiDAR and the ZED stereo camera got effected due to the dense fog, but the Navtech radar data remained largely the same.

A.2 The "Radiate" Test Vehicle

This section provides a detailed explanation of the sensor suite that was used to set up the test vehicle and its layout to collect good quality data.



(a) Foggy Day



(b) Clear Day

Figure A.2: Comparison between Velodyne LiDAR, Navtech Radar and ZED stereo camera in a foggy and clear day at the same location to understand the impact on all three sensor modalities in adverse weather condition.

Figure A.3: All four sensors used in our test vehicle. Starting from left CTS350-X Navtech Radar, ZED Stereo Camera, Velodyne HDL-32E LiDAR and Advance Navigation Spatial Dual Kit.

A.2.1 Sensor Specifications

The test vehicle sensor suite consists of four different sensor modalities illustrated in Figure. A.3, are as follows

- One ZED Stereo Camera to produce (672x376 pixel) left and right images at 30 Hz. It has two lenses of 2.8mm focal-length placed 120mm apart as a stereo baseline with a depth range from 0.5m to 25m and electronically synchronized rolling shutter.

- One Velodyne HDL-32E rotating 3D laser scanner with a range of 80-100m. Range Accuracy: $\pm 2cm$, Field of View: +10.67° to -30.67° (41.33°) vertical, 360° horizontal, Angular Resolution: 1.33° vertical, 0.1° to 0.4° horizontal and Rotation Rate: 5-20 Hz.

- One Spatial Dual Advance Navigation GPS/IMU kit that consists of temperature calibrated accelerometers, gyroscopes, magnetometers and a pressure sensor with a dual antenna Real-Time Kinematic (RTK) Global Navigation Satellite System (GNSS) receiver. The main reason behind the usage of two antennas, *i.e.* primary and secondary is to estimate the vehicle orientation/heading. Individual sensors are coupled with a sophisticated fusion algorithm to produce reliable and accurate navigation plus orientation. Horizontal Position Accuracy: 1.2m, Vertical Position Accuracy: 2.0m, Velocity Accuracy: 0.007m/s, Roll & Pitch Accuracy: 0.1°, Heading Accuracy at 1m Antenna Separation: 0.1°.

- One Navtech Radar CTS350-X with range 100m (radius) operating at $76-77$ GHz. Range Resolution: 0.175m, Azimuth and Elevation Beamwidth 1.8°, Field of View: 360°, Frame Rate: 4 Hz and Power Supply: 24V DC.

A.2.2 Sensor Positioning

The actual placement of our individual sensors onto our test vehicle (VW Transporter) is shown in Figure. A.1. To have the best and maximum field of view for all three perception sensor modules, i.e. LiDAR, radar and stereo camera, these were placed at the top-front of the vehicle. Since both the LiDAR and radar have a 360° field of view, they were placed at the front-right and front-left of the test vehicle respectively, keeping the centre open for the stereo-camera. For the GPS/IMU tool kit to achieve maximum GPS signal strength both the RTK GNSS receiver antennas (primary,secondary) were placed at the top of the van, of which the primary receiver was placed at the front and the secondary antenna at the rear with $1m$ separation to get the maximum heading accuracy. The IMU unit was placed on the floor of the car to alleviate the effects of the greater vibrations that usually occur at the top of the vehicle.

A.3 "Radiate" Software Architecture

In this section we discuss the developed software framework that was used to collect, process and store data in a meaningful folder structure categorized both in terms of the different sensors as well as the different scenarios.

A.3.1 Introduction of Robot Operating System

Our complete data collection software architecture was developed in the Robot Operating System framework. The Robot Operating System, or simply ROS, is an open-source meta-operating system which provides an interface between application and hardware. In our case the hardware consists of the various sensor modules mentioned in the previous section and the main task is to collect raw data from all these sensors and store it on the hard-disk.

Due to its loosely coupled design and the huge developers' community from all over the world working on various sensors and robot drivers, this has allowed researchers to reuse off the shelf sensor drivers in their own software pipeline instead of wasting time on "reinventing the wheel". Hardware abstraction, implementation of commonly-used functionality, low-level device control, package management and message-passing between processes are the five major tasks handled by this meta-OS. In the ROS framework processes are usually called as nodes and every node is responsible for one single task. Communications between nodes are usually done using message passing over pre-defined logical channels called topics. The nodes can be classified into two sub-categories, talker and listener, based on the task they are performing. The main responsibilities of each kind of node are as follows:

- **Talker:** A talker usually works as a sensor-driver which is responsible to communicate with the hardware/sensor to get the raw data, format it as a data message (usually called "sensor_name.msg") and send it over the topic.
- **Listener:** A listener usually listens into one or more topics and when there are any messages available in the topic it collects them and acts according to the requirement, either storage or further processing.

The data communication between talker and listener nodes is usually referred to as "publisher-subscriber" model where the talker is publishing sensor data over a topic and the listener is subscribing to the same topic to receive the data. In addition to these mentioned functionalities ROS also provides three visualization tools which are Gazebo, Rviz and MapViz to visualize the incoming data quickly and "bag" files to store data in the hard disk efficiently.

A.3.2 Architectural details

As our test vehicle consists of four different sensor modules, we created four different talker/publisher nodes as each sensor's driver, one listener/subscriber node to collect data from all the four talker nodes and save it in a bag file for later use (see Figure. A.4). Each talker node has its own name, specific message formats with the relevant data fields depending on the sensor, and is responsible for predefined topics to publish that message. Each sensor specific node and the associated topics and message file formats are given in Table A.1. After all the five nodes were developed we created a launch file to start all the nodes simultaneously instead of sequentially, thus reducing any potential risk of forgetting to start one or more nodes during real-time data collection.

A.3.3 Data Storage

Once the sequence specific sensor data was collected and stored in the bag files our next task was to properly extract individual raw sensor data and store it in a meaningful folder structure. A detailed structure of the data extraction stages are shown in Figure. A.5. Individual bags are usually of length 30-45 minutes. As it is hard to work with these giant bags we first split it into small bags of length 3 minutes each before extracting the raw sensor data. Moreover, splitting each bag into small bags will allow other users to select a limited amount of data from different sequences/scenarios (rain, snow, motorway, junction etc.) and thus prevent any potential bias that may occur due to a particular scenario. The split bags were named based on their master bag, which means if a bag is named as "sequence.bag" (sequence $\in$ rain, snow, fog, motorway, city, junction) the split bags were named

Table A.1: Details about the ROS framework used to collect data from all four sensors, i.e the nodes connected to each sensor, name of the topic data being published by the nodes and sensor message types along with the individual fields. In camera info messages, D, K, R and P represent the distortion parameters, intrinsic, rectification and projection matrix respectively. ROI is the region of interest and binning refers to any camera setting which combines rectangular neighborhoods of pixels into larger "super-pixels". In PointCloud2 message row and point steps were used jointly to separate points from each channel among 32 channels of Velodyne LiDAR.

Sensor	**Node**	**Topics**	**Message**	**Fields**
Radar	**navtech_ node**	/Navtech/Polar	sensor_msgs/Image.msg	height, width, encoding, is_bigendian, step, data
		/Navtech/Cartesian		
GPS/ IMU	**AND_ node**	/and/Imu	sensor_msgs/Imu.msg	orientation, orientation_covariance, angular_velocity, angular_velocity_covariance, linear_acceleration,linear_acceleration_covariance
		/and/NavSatFix	sensor_msgs/ NavSatFix.msg	latitude, longitude, altitude, position_covariance, position_covariance_type
		/and/Twist	geometry_msgs/Twist.msg	linear.x, linear.y, linear.z angular.x, angular.y, angular.z
		/and/Odom	nav_msgs/Odometry.msg	child_frame_id, pose, twist
Camera	**ZED_ node**	/left/info	sensor_msgs/ CameraInfo.msg	height, width, D, K, R, P, binning_x, binning_y, roi
		/right/info		
		/left/image	sensor_msgs/Image.msg	height, width, encoding, is_bigendian, step, data
		/right/image		
Lidar	**velo_ node**	/velodyne_points	sensor_msgs/ PointCloud2.msg	height, width, fields, is_bigendian, point_step, row_step, data, is_dense

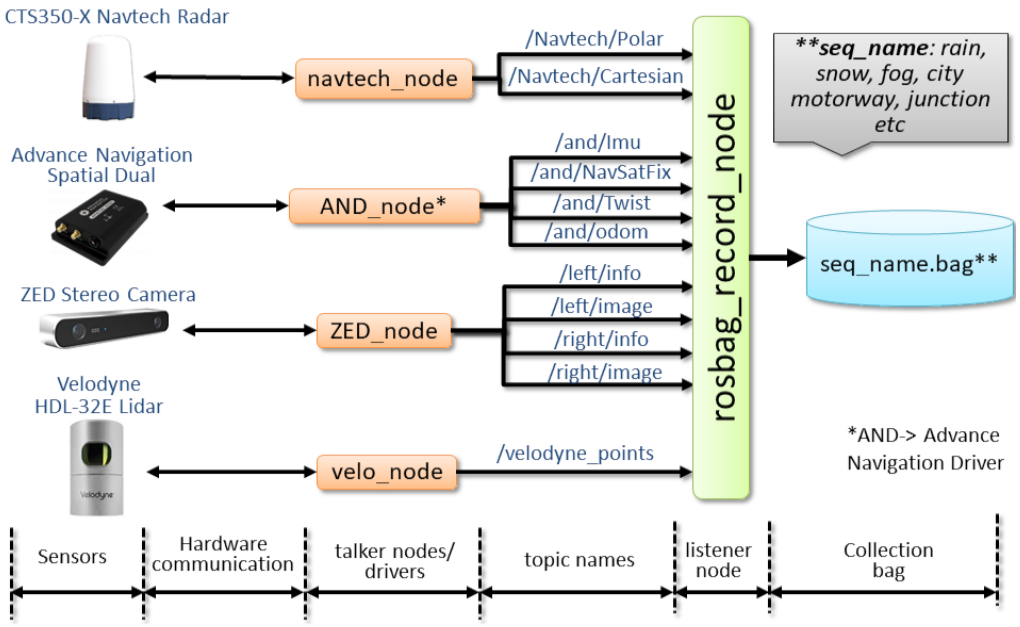


Figure A.4: End-to-end data collection pipeline developed using the Robot Operating System (ROS) framework, starting from the individual sensor drivers till storing the collected data into bag files.

as "sequence_1.bag" (0-3 min), "sequence_2.bag" (3-6 min), ... , "sequence_N.bag" (b-B min), where B is the length of the master bag and $b = 3 * (N - 1)$. Once the bags were split, each sensor's raw data was extracted into separate folders along with an additional file to save the timestamps of each frame. Timestamps were later used to synchronize the sensors as each sensor works at different frame rates.

A.4 "Radiate" Collection Sites

In order to cover different traffic scenarios the data collection was done in two different settings, i.e. static and moving ego-vehicle. The ego-vehicle is our sensor equipped test-vehicle shown in Figure A.1. The moving ego-vehicle scenario is for different highways where both the ego-vehicle and the surrounding vehicles were moving at different speeds. The static ego-vehicle scenario is for different unsignalled traffic intersections where the ego-vehicle was parked at a safe location near two different intersections to collect junction specific data. The unsignalled intersection allows us to understand how different vehicles will negotiate with each other with less mandatory influence from the traffic infrastructure. Different collection sites for both the static and moving ego-vehicle cases are explained in the following sections.

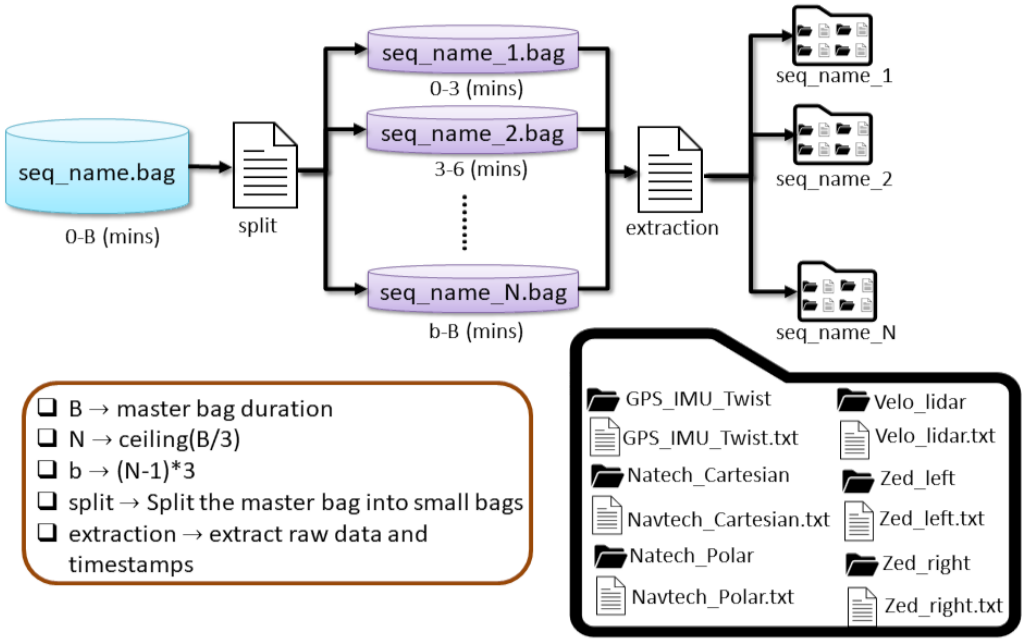


Figure A.5: Data extraction stages starting from the raw bag files produced by the subscriber node up to the individual sensor data stored into separate folders along with timestamps for each frame, categorized based on scenarios and split into small chunks of 3 minutes.

A.4.1 Moving Ego-Vehicle

The moving ego-vehicle scenario includes three different highways, i.e. single-lane, two-lane and three-lane motorway. The main goal of collecting highway data with three different lane counts is to understand its impact on high-speed driving behaviour by observing the lane-change events that happened during an overtake maneuver. In any two or three lane highway the main precursors for any overtaking maneuver are congestion in the ego-lane in which the vehicle is currently travelling, and availability in the adjacent lane to perform a safe overtake maneuver. Moreover, all nearby vehicles need to be considered first before starting the overtake maneuver and since all these vehicles will be travelling in the same direction it is likely the relative speeds between them are comparatively low. Given the correlating features and the lower relative speeds, it is possible to anticipate any future movement with a limited range on-board sensor, which is the "Navtech Radar" in our case with a range of 100 meters. On the other hand an overtaking maneuver performed on an single-lane motorway is a lot more critical as the over-taking vehicle will have to go to the opposing lane, in which the relevant vehicles' are travelling in the opposite direction, and with a very high relative speed. This is why to perform successful movement anticipation in a single-lane road the ego-vehicle must be equipped with sensors with higher distance range in order to cope with the higher relative speed. In some cases, while deciding the moving ego-vehicle's travel path, we kept the

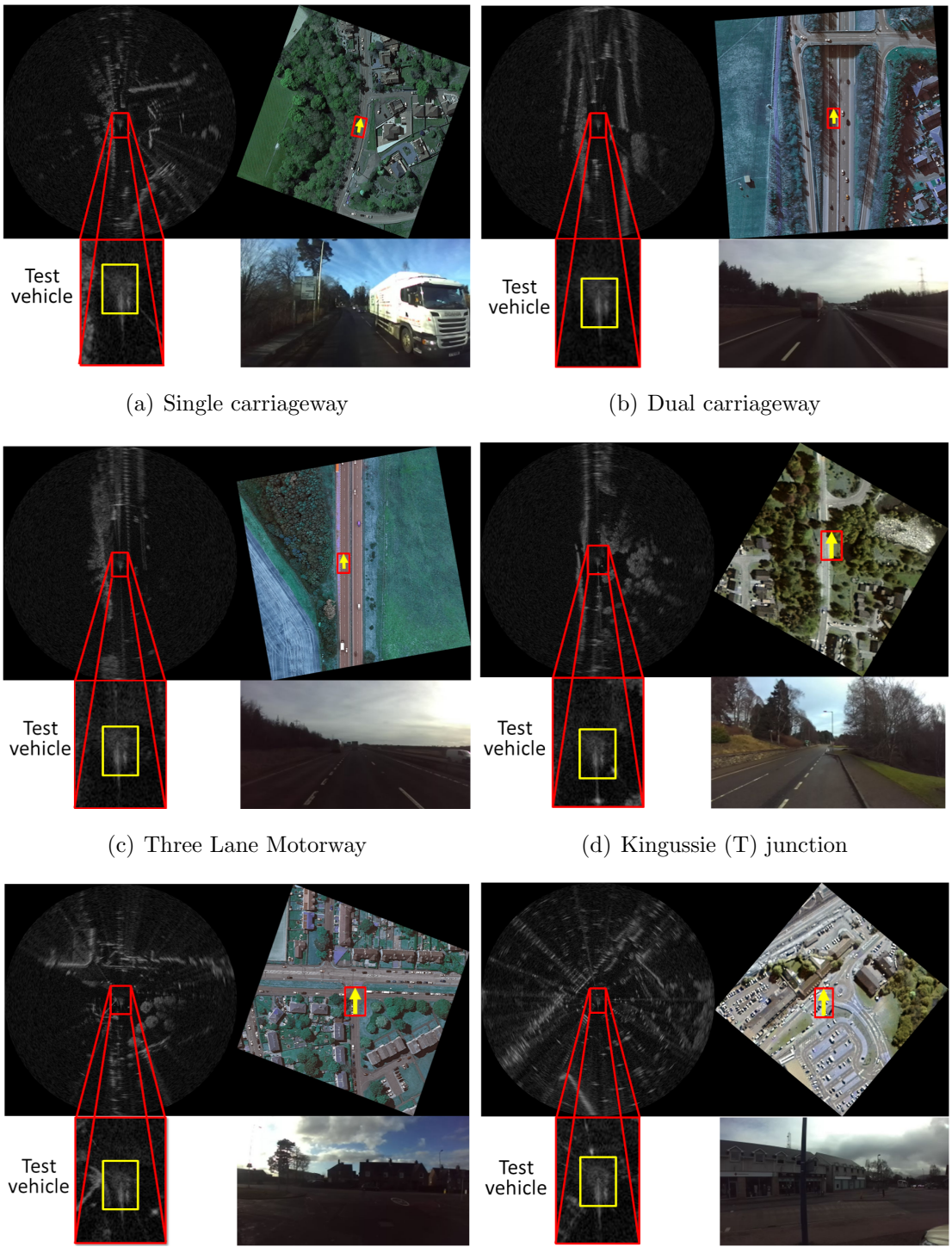
origin and destination locations the same so that the Visual-SLAM (Simultaneous Localization and Mapping) [72] community can also use this data with the loop-closure facility. Snapshots of the collected data from a single, two and three lane carriageways are shown in Figure A.6(a), A.6(b) and A.6(c) respectively.

A.4.2 Static Ego-Vehicle

The static ego-vehicle scenario includes three different types of intersections, i.e. T-Junction, four-Way junction and roundabout. In all three cases we parked our vehicle as close as possible to the junction and collected data. The main purpose behind collecting this static ego-vehicle data is to understand driving behaviour prior to different maneuvers that can be performed at any intersection such as left-turn, right-turn, straight-on etc. Moreover, collected vehicle data in these three different types of intersections can cover driving behaviour in most of the real world traffic junctions. The 100 meter range of our on-board Navtech radar system allowed us to see vehicles early enough so that past vehicle trajectories can be differentiated against distinct intersection specific maneuvers. The bird's-eye view (BEV) of all the three intersections along with the location of the parked test-vehicle are shown in Figure A.6(d), A.6(e) and A.6(f), respectively.

A.5 Pseudo codes of the Developed Algorithms

This section provides the developed pseudo codes of both the sequential prediction techniques explained in chapter 3 and 4. Algorithm 1 describes how the sequential prediction has been performed using the multi channel OGMs in chapter 3 and Algorithm 2 shows how those multi channels OGMs were created using the vehicles' position information. Next to that Algorithm 3 explains how the features vectors were populated using information from both the target vehicles as well as the dynamically selected surrounding vehicles in chapter 4 and Algorithm 4 shows how those features vectors were passed to the pre-trained model to perform the interactive prediction for all the vehicles currently present in the scene.



(a) Single carriageway

(b) Dual carriageway

(c) Three Lane Motorway

(d) Kingussie (T) junction

(e) Edinburgh (four-way) junction

(f) Kingussie (roundabout) junction

Figure A.6: Intersection specific static ego vehicle data collection. For each image top left image shows an example radar image collected with the navtech automotive radar. Red box indicates the location of the test vehicle near the junction. Top right image shows the satellite view of the junction [108], [110] and the yellow arrow indicates the direction of the test vehicle during data collection. Bottom right image is a sample image collected with the ZED Stereo Camera (RGB) during collection.

Result: TPred[N,F], predicted trajectory $\forall$ N

/*Past trajectory information*/

T[N,h] = $(x, y, Len, Wid)^{(1,N)}_{(t-h,t)}$ $\forall$ N ;

/*Store predicted trajectory*/

TPred[N,F] = $(x, y)^{(1,N)}_{(t,t+F)}$ $\forall$ N ;

for f in (t+1,t+F) **do**

  /*Create blank *inputOGMs* */

  inputOGMs[N,R,C,2,h] = 0;

  /*Populate *inputOGMs* considering each car as target $\forall$ N */

  for i in (1,N) **do**

    (orignX,originY) = (T[i,t-h].x,T[i,t-h].y);

    for j in (t-h,t) **do**

      (OGMX,OGMY) = (T[i,j].x-orignX)*LatRes,(T[i,j].y-orignY)*LonRes;

      (OGMWid,OGMLen) = (T[i,j].Wid*LatRes,T[i,j].Len*LonRes);

      /*Adding target car to inputOGMs channel 1*/

      inputOGMs[i,OGMX:OGMWid,OGMY:OGMLen,1,j] = 1;

      for s in (0,N) **do**

        /*Ignore the considered target car*/

        if s == i **then**

          continue;

        end

        (surrX,surrY) = (T[s,j].x-orignX)*LatRes,(T[s,j].y-originY)*LonRes;

        (surrWid,surrLen) = (T[s,j].Wid*LatRes,T[s,j].Len*LonRes);

        /*Adding surrounding car to OGM channel 2*/

        inputOGMs[i,surrX:surrWid,surrY:surrLen,2,j] = 1;

      end

    end

  end

  /*Predict next pose $(\hat{x}, \hat{y})$ $\forall$ N and add it to TPred[N,F] */

  for n in (1,N) **do**

    $(\hat{x},\hat{y})^n_f = model(inputOGMs[n, :, :, :, :])$;

    TPred[n,f] = $(\hat{x},\hat{y})^n_f$;

  end

  /*Adding the current predicted positions $\forall$ N to T */

  for k in (1,N) **do**

    /*Update the current trajectory for kth vehicle in T */

    xShift^k, $yShift^k = (T[k, t-h+1].x - T[k, t-h].x), (T[k, t-h+1].y - T[k, t-h].y)$;

    for l in (t-h+1,t) **do**

      T[k,l-1].x,T[k,l-1].y = (T[k,l].x-xShift^k), $(T[k, l].y - yShift^k)$;

    end

    /*Add the current predicted pose of kth vehicle in T */

    T[k,t].x,T[k,t].y = (TPred[k,f].$\hat{x}$),(TPred[k,f].$\hat{y}$);

  end

end

Algorithm 1: Pseudocode for future trajectory prediction of all the cars (N) present in the scene for the next **F** future frames given h historical frames, where *model* is the pre-trained model. All (N) vehicles must be present in the entire observation sequence i.e. from $(t - h)$ to (t)

```
Result: (inputOGMs,outputPosXY) ∀ training and validation samples
(LatRes,LonRes) = (C/GirdWid, R/GirdLen);
CarDict = dict(CarID, [x,y,Len,Wid,FrameID] ∀ FrameID of CarID);
FrameDict = dict(FrameID), [x,y,Len,Wid,carID] ∀ CarID at FrameID;
for CarID in CarDict do
    (targetCarID,TargetList) = (CarID, CarDict[CarID]);
    TargetLen = length(TargetList);
    for i in (0,TargetLen-h-1) do
        /*Create blank inputOGMs*/
        inputOGMs[R,C,2,h] = 0;
        (originX,originY) = (TargetList[i].x,TargetList[i].y);
        for j in (i,i+h) do
            (x,y,Len,Wid,currentFrame) = TargetList[j];
            (relativeX,relativeY) = (x-originX,y-originY);
            /*Create blank OGM*/
            OGM[R,C,2] = 0;
            (OGMX,OGMY)= (relativeX*LatRes,relativeY*LonRes);
            (OGMWid,OGMLen) = (Wid*LatRes,Len*LonRes);
            /*Adding Target Car to first OGM channel*/
            inputOGMs[OGMX:OGMWid,OGMY:OGMLen,1,j] = 255;
            SurroundingCars = FrameDict[currentFrame];
            for eachSurrCar in SurroundingCars do
                (surrX,surrY,surrLen,surrWid,surrId) = eachSurrCar;
                if surrId == targetCarID then
                    /*Target Car is already added*/
                    continue;
                end
                (surrRelX,surrRelY) = (surrX-originX,surrY-originY);
                (OGMX,OGMY) = (surrRelX*LatRes, surrRelY*LonRes);
                (OGMWid,OGMLen) = (surrWid*LatRes, surrLen*LonRes);
                /*Adding Surrounding Car to second OGM channel*/
                inputOGMs[OGMX:OGMWid,OGMY:OGMLen,2,j] = 255;
            end
        end
        (outPosX,outPosY) = (TargetList[i+h+1].x,TargetList[i+h+1].y);
        (relativeOutX,relativeOutY) = (x-outPosX, y-outPosY);
        outputPoseXY = [relativeOutX,relativeOutY];
        Add (inputOGMs,outputPoseXY) to sampleList;
    end
end
```

Algorithm 2: Occupancy Grid Map (OGM) generation pseudocode with respect to each car in the scene for model training. *GridLen* and $GridWid$ are the physical length and width of the grid in the real world measured in feet. R and C are the number of rows and columns of the generated input OGMs and *dict* is dictionary. *sampleList* is the list of samples generated by the OGM sample generation algorithm and is used for training and validation of the proposed network.

Result: $\mathbf{O}^{(1,N)}_{(t-h,t)}$, $\mathbf{D}^{(1,N)}_{(t+1)}$
/*Past trajectory information with map features*/
T[N,h] = $(x, y, \dot{x}, \dot{y}, \beta, j, d, \mu)^{1,N}_{t,t-h}$;
/*Store Predicted $\hat{\mathbf{m}}$, $(\hat{\hat{x}}, \hat{\hat{y}})$ and $(\hat{x}, \hat{y})$ $\forall$ N */
TPred[N,F] = $[\hat{\mathbf{m}}, (\hat{\hat{x}}, \hat{\hat{y}}), (\hat{x}, \hat{y})]^{(1,N)}_{(t+1,t+F)}$
/*Create input feature vector $\mathbf{O}^{1,N}_{t-h,t}$ */
for i in (1, N) **do**
  for j in (t-h, t) **do**
    /*Add Target Vehicle TV*/
    $\mathbf{TV}_i = T[i, j].[x, y, \dot{x}, \dot{y}, \beta, j, d, \mu]$;
    $\mathbf{O}^i_j = [\mathbf{TV}_i]$;
    /*Select most influential SVs*/
    DistArray[N-1,2] = $[\text{EDist}_d, d], d \; \forall$ N : d $\neq i$;
    /*Estimate Euclidean distance between TV and all SVs*/
    for d in (1, N) **do**
      /*Ignore Target Vehicle (TV)*/
      if d == i **then**
        continue;
      end
      $\text{EDist}_d = Euclidean[T[i, j].(x, y), T[d, j].(x, y)]$;
      DistArray[d] = $[\text{EDist}_d, d]$;
    end
    /*Sort *DistArray* by Column 1*/
    sort($DistArray$, 1);
    /*Add selected SVs*/
    for l in (1:ns) **do**
      SVId = DistArray[l,2];
      $\mathbf{SV}_l = [T[SVId, j].[x, y, \dot{x}, \dot{y}, \beta, j, d, \mu]]$;
      $\mathbf{O}^i_j = [\mathbf{O}^i_j, \mathbf{SV}_l]$;
    end
  end
end
/* Populate initial D_{t+1} $\forall$ N */
for k in (1,N) **do**
  $\mathbf{m}, (\dot{x}, \dot{y}), (x, y)$ = HotVector($\mathbf{O}^k_t.\mathbf{TV}_k.\mu$), ($\mathbf{O}^k_t.\mathbf{TV}_k.(\dot{x}, \dot{y})$), ($\mathbf{O}^k_t.\mathbf{TV}_k.(x, y)$);
  $\mathbf{D}^k_{t+1} = [\mathbf{m}, (\dot{x}, \dot{y}), (x, y)]$;
  for l = 1:ns **do**
    $\mathbf{m}, (\dot{x}, \dot{y}), (x, y)$ = HotVector($\mathbf{O}^k_t.SV_l.\mu$), ($\mathbf{O}^k_t.\mathbf{SV}_l.(\dot{x}, \dot{y})$), ($\mathbf{O}^k_t.\mathbf{SV}_l.(x, y)$);
    $\mathbf{D}^k_{t+1} = [\mathbf{D}^k_{t+1}, \mathbf{m}, (\dot{x}, \dot{y}), (x, y)]$;
  end
end

Algorithm 3: Input feature vector $\mathbf{O}^{1,N}_{t-h,t}$ and initial decoder input $\mathbf{D}^{1,N}_{(t+1)}$ creation for all the N cars present in the scene. Both $\mathbf{O}^{1,N}_{t-h,t}$ and $\mathbf{D}^{1,N}_{(t+1)}$ will be used by the sequential interactive prediction technique explained in the next algorithm. $EDist_d$ is the Euclidean distance between the current Target Vehicle (TV) and dth Surrounding Vehicle (SV^d)

Result: $[\hat{\mathbf{m}}, (\hat{\hat{x}}, \hat{\hat{y}}), (\hat{x}, \hat{y})]^{(1,N)}_{(t+1,t+F)}$
/*Start sequential prediction*/
/*Store Encoded state for all vehicles*/
EncodeArray[N] = $(\mathrm{ES}_1, ES_2)^{(1,N)}$;
for i in (1,N) **do**
    $\mathrm{ES}^i_1, ES^i_2 = Encoder(\mathbf{O}^i_{t-h,t})$;
    EncodeArray[i] = $[\mathrm{ES}^i_1, ES^i_2]$;
end
for f in (t+1,t+F) **do**
    /*Predict next frame ∀ N */
    for i in (1,N)) **do**
        $[\hat{\mathbf{m}}, (\hat{\hat{x}}, \hat{\hat{y}}), (\hat{x}, \hat{y})]^i_f, [ES_1, ES_2]^i = Decode(D^i_f, EncodeArray[i])$;
        /*Update state for ith vehicle */
        EncodeArray[i] = $[\mathrm{ES}_1, ES_2]^i$;
        /*Store predicted entities for ith vehicle at fth time */
        TPred[i,f] = $[\hat{\mathbf{m}}, (\hat{\hat{x}}, \hat{\hat{y}}), (\hat{x}, \hat{y})]^i_f$;
    end
    /*Populate $\mathrm{D}_{f+1}\forall$ N with predicted entities*/
    for j in (1,N) **do**
        /*Populate D^j_f with TV Features*/
        $\mathrm{D}^j_{f+1} = TPred[j, f]$;
        /*Populate D^j_{f+1} with SV Features*/
        /*Estimate Euclidean distance between predicted TV and ∀ N ≠ j */
        DistArray[N-1,2] = $[\mathrm{EDist}_d, d], \forall d \in N : d \neq j$;
        for d in (1, N) **do**
            /*Ignore Target Vehicle (TV)*/
            if d == j **then**
                continue;
            end
            $\mathrm{EDist}_d = Euclidean[TPred[j, f].(\hat{x}, \hat{y}), TPred[d, f].(\hat{x}, \hat{y})]$;
            DistArray[d] = $[\mathrm{EDist}_d, d]$;
        end
        /*Sort *DistArray* by Column 1 */
        sort(*DistArray*, 1);
        /*Add selected SVs to D^j_{f+1} */
        for l in (1:ns) **do**
            SVId = DistArray[l,2];
            SVPred = TPred[l,f].$[\hat{\mathbf{m}}, (\hat{\hat{x}}, \hat{\hat{y}}), (\hat{x}, \hat{y})]$;
            $\mathbf{D}^j_{f+1} = [\mathbf{D}^j_{f+1}, \mathbf{SVPred}]$;
        end
    end
end

Algorithm 4: Interactive prediction technique for future position of all the cars for the next **F** future frames given h historical frames. ES_1 and ES_2 are encoded state from stacked LSTM_1 and LSTM_2 respectively. $EDist_d$ is the Euclidean distance between the current Target Vehicle (TV) and dth Surrounding Vehicle (SV^d)

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), pages 265–283, 2016.

[2] Florent Altché and Arnaud de La Fortelle. An lstm network for highway trajectory prediction. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pages 353–359. IEEE, 2017.

[3] Aleksandar Angelov, Andrew Robertson, Roderick Murray-Smith, and Francesco Fioranelli. Practical classification of different moving targets using automotive radar and deep neural networks. IET Radar, Sonar & Navigation, 12(10):1082–1089, 2018.

[4] Martin Arjovsky and Soumith Chintala. Bottou. wasserstein gan. arXiv preprint arXiv:1701.07875, 7, 2017.

[5] David Augustin, Marius Hofmann, and Ulrich Konigorski. Motion pattern recognition for maneuver detection and trajectory prediction on highways. In 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES), pages 1–8. IEEE, 2018.

[6] David Augustin, Marius Hofmann, and Ulrich Konigorski. Prediction of highway lane changes based on prototype trajectories. Forschung im Ingenieurwesen, 83(2):149–161, 2019.

[7] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Meireles Paixao, Filipe Mutz, et al. Self-driving cars: A survey. Expert Systems with Applications, page 113816, 2020.

[8] Minjin Baek, Donggi Jeong, Dongho Choi, and Sangsun Lee. Vehicle trajectory prediction and collision warning via fusion of multisensors and wireless vehicular communications. Sensors, 20(1):288, 2020.

[9] Jawadul H Bappy, Cody Simons, Lakshmanan Nataraj, BS Manjunath, and Amit K Roy-Chowdhury. Hybrid lstm and encoder–decoder architecture for detection of image forgeries. IEEE Transactions on Image Processing, 28(7):3286–3300, 2019.

[10] Dan Barnes, Matthew Gadd, Paul Murcutt, Paul Newman, and Ingmar Posner. The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 6433–6438. IEEE, 2020.

[11] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In Advances in Neural Information Processing Systems, pages 1171–1179, 2015.

[12] J. V. D. Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In Robotics research, pages 3–19. Springer, 2011.

[13] S. Bonnin, T. H. Weisswange, F. Kummert, and J. Schmudderich. Accurate behavior prediction on highways based on a systematic combination of classifiers. In Intelligent Vehicles Symposium (IV), 2013 IEEE, pages 242–249. IEEE, 2013.

[14] Abderrahmane Boubezoul, Abdourahmane Koita, and Dimitri Daucher. Vehicle trajectories classification using support vectors machines for failure trajectory prediction. In 2009 International Conference on Advances in Computational Tools for Engineering Applications, pages 486–491. IEEE, 2009.

[15] Yuheng Bu, Shaofeng Zou, Yingbin Liang, and Venugopal V Veeravalli. Estimation of kl divergence: Optimal minimax rate. IEEE Transactions on Information Theory, 64(4):2648–2674, 2018.

[16] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027, 2019.

[17] Wenqi Cai, Ganglei He, Jianlong Hu, Haiyan Zhao, Yuhai Wang, and Bingzhao Gao. A comprehensive intention prediction method considering vehicle interaction. In 2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI), pages 204–209. IEEE, 2020.

[18] Xiaomeng Cao, Jian Lan, X Rong Li, and Yu Liu. Extended object tracking using automotive radar. In 2018 21st International Conference on Information Fusion (FUSION), pages 1–5. IEEE, 2018.

[19] Yaron Caspi and Michal Irani. A step towards sequence-to-sequence alignment. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662), volume 2, pages 682–689. IEEE, 2000.

[20] José Castanheira, Francisco Curado, Ana Tomé, and Edgar Gonçalves. Machine learning methods for radar-based people detection and tracking. In EPIA Conference on Artificial Intelligence, pages 412–423. Springer, 2019.

[21] S. Chen. Kalman filter for robot vision: a survey. IEEE Transactions on Industrial Electronics, 59(11):4409–4420, 2011.

[22] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

[23] J. Colyar and J. Halkias. US highway 101 dataset. In Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030, 2007.

[24] J. Colyar and J. Halkias. US highway I-80 dataset. In Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030, 2007.

[25] J. Colyar and J. Halkias. US highway Lankershim Boulevard dataset. In Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030, 2007.

[26] J. Colyar and J. Halkias. US highway Peachtree Street dataset. In Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030, 2007.

[27] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3213–3223, 2016.

[28] S. Dai, L. Li, and Z. Li. Modeling vehicle interactions via modified lstm models for trajectory prediction. IEEE Access, 7:38287–38296, 2019.

[29] N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1468–1476, 2018.

[30] Nachiket Deo, Akshay Rangesh, and Mohan M Trivedi. How would surround vehicles move? a unified framework for maneuver classification and motion prediction. IEEE Transactions on Intelligent Vehicles, 3(2):129–140, 2018.

[31] Nachiket Deo and Mohan M Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 1179–1184. IEEE, 2018.

[32] Oussama Derbel, Tamas Peter, Hossni Zebiri, Benjamin Mourllion, and Michel Basset. Modified intelligent driver model for driver safety and traffic stability improvement. IFAC Proceedings Volumes, 46(21):744–749, 2013.

[33] Jacob Devlin, Rudy Bunel, Rishabh Singh, Matthew Hausknecht, and Pushmeet Kohli. Neural program meta-induction. In Advances in Neural Information Processing Systems, page 2080–2088, 2017.

[34] Juergen Dickmann, Jens Klappstein, Markus Hahn, Nils Appenrodt, Hans-Ludwig Bloecher, Klaudius Werber, and Alfons Sailer. Automotive radar the key technology for autonomous driving: From detection and ranging to environmental understanding. In 2016 IEEE Radar Conference (RadarConf), pages 1–6. IEEE, 2016.

[35] Chiyu Dong, Yilun Chen, and John M Dolan. Interactive trajectory prediction for autonomous driving via recurrent meta induction neural network. In 2019 International Conference on Robotics and Automation (ICRA), pages 1212–1217. IEEE, 2019.

[36] Xu Dong, Pengluo Wang, Pengyue Zhang, and Langechuan Liu. Probabilistic oriented object detection in automotive radar. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 102–103, 2020.

[37] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In Conference on robot learning, pages 1–16. PMLR, 2017.

[38] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[39] Hao Du, Beibei Ge, Yongpeng Dai, and Tian Jin. Knowing the uncertainty in human behavior classification via variational inference and autoencoder. In 2019 International Radar Conference (RADAR), pages 1–4. IEEE, 2019.

[40] Xunsheng Du, Huaqing Zhang, Hien Van Nguyen, and Zhu Han. Stacked lstm deep learning model for traffic prediction in vehicle-to-vehicle communication. In 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall), pages 1–5. IEEE, 2017.

[41] Rob A Dunne and Norm A Campbell. On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In Proc. 8th Aust. Conf. on the Neural Networks, Melbourne, volume 181, page 185. Citeseer, 1997.

[42] Mica R Endsley. Autonomous driving systems: A preliminary naturalistic study of the tesla model s. Journal of Cognitive Engineering and Decision Making, 11(3):225–238, 2017.

[43] Florian Engels, Philipp Heidenreich, Abdelhak M Zoubir, Friedrich K Jondral, and Markus Wintermantel. Advances in automotive radar: A framework on computationally efficient high-resolution frequency estimation. IEEE Signal Processing Magazine, 34(2):36–46, 2017.

[44] Cong Fei, Xiangkun He, and Xuewu Ji. Multi-modal vehicle trajectory prediction based on mutual information. IET Intelligent Transport Systems, 14(3):148–153, 2019.

[45] Cong Fei, Xiangkun He, Sadahiro Kawahara, Nakano Shirou, and Xuewu Ji. Conditional wasserstein auto-encoder for interactive vehicle trajectory prediction. In 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pages 1–6. IEEE, 2020.

[46] Zhaofei Feng. Lane and Road Marking Detection with a High Resolution Automotive Radar for Automated Driving. PhD thesis, KIT-Bibliothek, 2019.

[47] Zhaofei Feng, Martin Stolz, Mingkang Li, Martin Kunert, and Werner Wiesbeck. Verification of a lane detection method with automotive radar based on a new type of road marking. In 2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), pages 1–4. IEEE, 2018.

[48] Florian Folster, Hermann Rohling, and Urs Lubbert. An automotive radar network based on 77 ghz fmcw sensors. In IEEE International Radar Conference, 2005., pages 871–876. IEEE, 2005.

[49] Christian-Eike Framing, Frank-Josef Heßeler, and Dirk Abel. Infrastructure-based vehicle maneuver estimation with intersection-specific models. In 2018 26th Mediterranean Conference on Control and Automation (MED), pages 253–258. IEEE, 2018.

[50] V. Gadepally, A. Krishnamurthy, and U. Ozguner. A framework for estimating driver decisions near intersections. IEEE Transactions on Intelligent Transportation Systems, 15(2):637–646, 2014.

[51] V. Gadepally, A. Krishnamurthy, and Ü. Özgüner. A framework for estimating long term driver behavior. Journal of Advanced Transportation, 2017, 2017.

[52] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In International Conference on Machine Learning, pages 1704–1713. PMLR, 2018.

[53] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. The International Journal of Robotics Research, 32(11):1231–1237, 2013.

[54] X. Geng, H. Liang, B. Yu, P Zhao, L. He, and R. Huang. A scenario-adaptive driving behavior prediction approach to urban autonomous driving. Applied Sciences, 7(4):426, 2017.

[55] Samuel Gibbs. Google sibling waymo launches fully autonomous ride-hailing service. The Guardian, 7, 2017.

[56] Simon Godsill. Particle filtering: the first 25 years and beyond. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7760–7764. IEEE, 2019.

[57] A. Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[58] Nathan A Greenblatt. Self-driving cars and the law. IEEE spectrum, 53(2):46–51, 2016.

[59] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. Journal of Field Robotics, 37(3):362–386, 2020.

[60] Dale M Grimes and Trevor Owen Jones. Automotive radar: A brief review. Proceedings of the IEEE, 62(6):804–822, 1974.

[61] Christopher Grimm, Ridha Farhoud, Tai Fei, Ernst Warsitz, and Reinhold Haeb-Umbach. Detection of moving targets in automotive radar with distorted ego-velocity information. In 2017 IEEE Microwaves, Radar and Remote Sensing Symposium (MRRS), pages 111–116. IEEE, 2017.

[62] Xiaodong Gu, Kyunghyun Cho, Jung-Woo Ha, and Sunghun Kim. Dialogwae: Multimodal response generation with conditional wasserstein auto-encoder. arXiv preprint arXiv:1805.12352, 2018.

[63] Dong Guan, Hui Zhao, Long Zhao, and Kan Zheng. Intelligent prediction of mobile vehicle trajectory based on space-time information. In 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), pages 1–5. IEEE, 2019.

[64] Gor Hakobyan and Bin Yang. High-performance automotive radar: A review of signal processing algorithms and modulation schemes. IEEE Signal Processing Magazine, 36(5):32–44, 2019.

[65] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. Physical review E, 51(5):4282, 1995.

[66] Patrick Held, Dagmar Steinhauser, Alexander Kamann, Andreas Koch, Thomas Brandmeier, and Ulrich T Schwarz. Micro-doppler extraction of bicycle pedaling movements using automotive radar. In 2019 IEEE Intelligent Vehicles Symposium (IV), pages 744–749. IEEE, 2019.

[67] Patrick Held, Dagmar Steinhauser, Andreas Koch, Thomas Brandmeier, and Ulrich T Schwarz. A novel approach for model-based pedestrian tracking using automotive radar. IEEE Transactions on Intelligent Transportation Systems, 2021.

[68] A. Hermann and J. Desel. Driving situation analysis in automotive environment. In Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on, pages 216–221. IEEE, 2008.

[69] Edward G Hoare, PS Hall, R Hill, SH Tsang, C Thompson, S Fu, and N Clarke. Millimetre-wave automotive radar advance path measurement. Technical report, SAE Technical Paper, 2002.

[70] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[71] Stefan Hoermann, Martin Bach, and Klaus Dietmayer. Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with

fully automatic labeling. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 2056–2063. IEEE, 2018.

[72] Ziyang Hong, Yvan Petillot, and Sen Wang. Radarslam: Radar based large-scale slam in all weathers. arXiv preprint arXiv:2005.02198, 2020.

[73] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 79(8):2554–2558, 1982.

[74] Lian Hou, Long Xin, Shengbo Eben Li, Bo Cheng, and Wenjun Wang. Interactive trajectory prediction of surrounding road users for autonomous driving using structural-lstm network. IEEE Transactions on Intelligent Transportation Systems, 2019.

[75] Adam Houenou, Philippe Bonnifait, Véronique Cherfaoui, and Wen Yao. Vehicle trajectory prediction based on motion model and maneuver recognition. In 2013 IEEE/RSJ international conference on intelligent robots and systems, pages 4363–4369. IEEE, 2013.

[76] Tsung-Han Hsieh, Li Su, and Yi-Hsuan Yang. A streamlined encoder/decoder architecture for melody extraction. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 156–160. IEEE, 2019.

[77] Yeping Hu, Wei Zhan, and Masayoshi Tomizuka. Probabilistic prediction of vehicle semantic intention and motion. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 307–313. IEEE, 2018.

[78] Constantin Hubmann, Jens Schulz, Marvin Becker, Daniel Althoff, and Christoph Stiller. Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction. IEEE Transactions on Intelligent Vehicles, 3(1):5–17, 2018.

[79] Texas Instruments. Short range radar reference design using awr1642. Technical report, Technical Report, 2017.

[80] Ashesh Jain, Hema S Koppula, Bharad Raghavan, Shane Soh, and Ashutosh Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In Proceedings of the IEEE International Conference on Computer Vision, pages 3182–3190, 2015.

[81] Donya Jasteh. Experimental low-THz imaging radar for automotive applications. PhD thesis, University of Birmingham, 2017.

[82] H. S. Jeon, D. S. Kum, and W. Y. Jeong. Traffic scene prediction via deep learning: Introduction of multi-channel occupancy grid map as a scene representation. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 1496–1501. IEEE, 2018.

[83] Hyeongseok Jeon, Junwon Choi, and Dongsuk Kum. Scale-net: Scalable vehicle trajectory prediction network under random number of interacting vehicles via edge-enhanced graph convolutional neural network. arXiv preprint arXiv:2002.12609, 2020.

[84] Daejun Kang and Dongsuk Kum. Camera and radar sensor fusion for robust vehicle localization via vehicle part localization. IEEE Access, 8:75223–75236, 2020.

[85] N. Ketkar. Introduction to keras. In Deep Learning with Python, pages 97–111. Springer, 2017.

[86] Beomjun Kim, Kyongsu Yi, Hyun-Jae Yoo, Hyok-Jin Chong, and Bongchul Ko. An imm/ekf approach for enhanced multitarget state estimation for application to integrated risk management system. IEEE Transactions on Vehicular Technology, 64(3):876–889, 2014.

[87] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pages 399–404. IEEE, 2017.

[88] Jihun Kim, Žiga Emeršič, and Dong Seog Han. Vehicle path prediction based on radar and vision sensor fusion for safe lane changing. In 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), pages 267–271. IEEE, 2019.

[89] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein. The High-D 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), pages 2118–2125. IEEE, 2018.

[90] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier. Learning-based approach for online lane change intention prediction. In 2013 IEEE Intelligent Vehicles Symposium (IV), pages 797–802. IEEE, 2013.

[91] Y. M. Lee and E. Sheppard. The effect of motion and signalling on drivers' ability to predict intentions of other road users. Accident Analysis & Prevention, 95:202–208, 2016.

[92] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. Robomech Journal, 1(1):1, 2014.

[93] Junxiang Li, Bin Dai, Xiaohui Li, Xin Xu, and Daxue Liu. A dynamic bayesian network for vehicle maneuver prediction in highway driving scenarios: Framework and verification. Electronics, 8(1):40, 2019.

[94] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. Grip: Graph-based interaction-aware trajectory prediction. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pages 3960–3966. IEEE, 2019.

[95] Yang Li, Can Liang, Man Lu, Xueyao Hu, and Yanhua Wang. Cascaded kalman filter for target tracking in automotive radar. The Journal of Engineering, 2019(19):6264–6267, 2019.

[96] Martin Liebner, Michael Baumann, Felix Klanner, and Christoph Stiller. Driver intent inference at urban intersections using the intelligent driver model. In 2012 IEEE Intelligent Vehicles Symposium, pages 1162–1167. IEEE, 2012.

[97] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. Pattern recognition, 36(2):451–461, 2003.

[98] Pedro F Lima. Optimization-based motion planning and model predictive control for autonomous driving: With experimental evaluation on a heavy-duty construction truck. PhD thesis, KTH Royal Institute of Technology, 2018.

[99] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. arXiv preprint arXiv:1511.06114, 2015.

[100] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 6120–6127, 2019.

[101] Zhikai Ma, Qian Huo, Xin Yang, and Xiaoshun Zhao. Safety cruise control of connected vehicles using radar and vehicle-to-vehicle communication. IEEE Systems Journal, 2020.

[102] Adrian Macaveiu and Andrei Câmpeanu. Automotive radar target tracking by kalman filtering. In 2013 11th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services (TELSIKS), volume 2, pages 553–556. IEEE, 2013.

[103] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. The International Journal of Robotics Research, 36(1):3–15, 2017.

[104] Georgiana Magu, Radu Lucaciu, and Alexandru Isar. Improving the targets' trajectories estimated by an automotive radar sensor using polynomial fitting. Applied Sciences, 11(1):361, 2021.

[105] Bence Major, Daniel Fontijne, Amin Ansari, Ravi Teja Sukhavasi, Radhika Gowaikar, Michael Hamilton, Sean Lee, Slawomir Grzechnik, and Sundar Subramanian. Vehicle detection with automotive radar using deep learning on range-azimuth-doppler tensors. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, pages 0–0, 2019.

[106] H. M. Mandalia and M. D. D. Salvucci. Using support vector machines for lane-change detection. In Proceedings of the human factors and ergonomics society annual meeting, volume 49, pages 1965–1969. SAGE Publications Sage CA: Los Angeles, CA, 2005.

[107] J. Mänttäri, J. Folkesson, and E. Ward. Learning to predict lane changes in highway scenarios using dynamic filters on a generic traffic representation. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 1385–1392. IEEE, 2018.

[108] Google Maps. Edinburgh, Scotland, UK. maps.google.com, 2018.

[109] Google Maps. Karlsruhe, Germany. maps.google.com, 2018.

[110] Google Maps. Kingussie, Scotland, UK. maps.google.com, 2018.

[111] Google Maps. Lankershim USA. maps.google.com, 2018.

[112] David Meister, Martin F Holder, and Hermann Winner. A track-before-detect approach to multi-target tracking on automotive radar sensor data. arXiv preprint arXiv:2006.02755, 2020.

[113] P. Meletis and G. Dubbelman. Training of convolutional networks on multiple heterogeneous datasets for street scene semantic segmentation. IEEE Intelligent Vehicles Symposium (IV), 2018.

[114] Kaouther Messaoud, Itheri Yahiaoui, Anne Verroust-Blondet, and Fawzi Nashashibi. Relational recurrent neural networks for vehicle trajectory prediction. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pages 1813–1818. IEEE, 2019.

[115] N. Mohajerin and M. Rohani. Multi-step prediction of occupancy grid maps with recurrent neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 10600–10608, 2019.

[116] Todd K Moon. The expectation-maximization algorithm. IEEE Signal processing magazine, 13(6):47–60, 1996.

[117] Sajjad Mozaffari, Omar Y Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. IEEE Transactions on Intelligent Transportation Systems, 2020.

[118] Dominik Notz, Felix Becker, Thomas Kühbeck, and Daniel Watzenig. Extraction and assessment of naturalistic human driving trajectories from infrastructure camera and radar sensors. arXiv preprint arXiv:2004.01288, 2020.

[119] Masaki Okamoto, Pietro Perona, and Abdelaziz Khiat. Ddt: Deep driving tree for proactive planning in interactive scenarios. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 656–661. IEEE, 2018.

[120] Michaël Garcia Ortiz, Franz Kummert, and Jens Schmüdderich. Prediction of driver behavior on a limited sensory setting. In 2012 15th International IEEE Conference on Intelligent Transportation Systems, pages 638–643. IEEE, 2012.

[121] 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. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 1672–1678. IEEE, 2018.

[122] Rodrigo Pérez, Falk Schubert, Ralph Rasshofer, and Erwin Biebl. Deep learning radar object detection and classification for urban automotive scenarios. In 2019 Kleinheubach Conference, pages 1–4. IEEE, 2019.

[123] Derek J Phillips, Tim A Wheeler, and Mykel J Kochenderfer. Generalizable intention prediction of human drivers at intersections. In 2017 IEEE Intelligent Vehicles Symposium (IV), pages 1665–1670. IEEE, 2017.

[124] Aris Polychronopoulos, Manolis Tsogas, Angelos J Amditis, and Luisa Andreone. Sensor fusion for predicting vehicles' path for collision avoidance systems. IEEE Transactions on Intelligent Transportation Systems, 8(3):549–562, 2007.

[125] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In ICRA workshop on open source software, volume 3, page 5. Kobe, Japan, 2009.

[126] G. Raipuria, F. Gaisser, and P. P. Jonker. Road infrastructure indicators for trajectory prediction. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 537–543. IEEE, 2018.

[127] P. Ratsamee, Y. Mae, K. Ohara, T. Takubo, and T. Arai. Human–robot collision avoidance using a modified social force model with body pose and face orientation. International Journal of Humanoid Robotics, 10(01):1350008, 2013.

[128] L. Rittger, G. Schmidt, C. Maag, and A. Kiesel. Driving behaviour at traffic light intersections. Cognition, Technology & Work, 17(4):593–605, 2015.

[129] Kåre Rumar. Human factor in road safety. Statens Väg-och Trafikinstitut. VTI Särtryck 81, 1982.

[130] Eder Santana and George Hotz. Learning a driving simulator. arXiv preprint arXiv:1608.01230, 2016.

[131] Atri Sarkar, Krzysztof Czarnecki, Matt Angus, Changjian Li, and Steven Waslander. Trajectory prediction of traffic agents at urban intersections through learned interactions. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pages 1–8. IEEE, 2017.

[132] Nicolas Scheiner, Florian Kraus, Fangyin Wei, Buu Phan, Fahim Mannan, Nils Appenrodt, Werner Ritter, Jurgen Dickmann, Klaus Dietmayer, Bernhard Sick, et al. Seeing around street corners: Non-line-of-sight detection and tracking in-the-wild using doppler radar. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2068–2077, 2020.

[133] Julian Schlechtriemen, Florian Wirthmueller, Andreas Wedel, Gabi Breuel, and Klaus-Dieter Kuhnert. When will it change the lane? a probabilistic regression approach for rarely occurring events. In 2015 IEEE Intelligent Vehicles Symposium (IV), pages 1373–1379. IEEE, 2015.

[134] Matthias Schreier, Volker Willert, and Jürgen Adamy. Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems. In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on, pages 334–341. IEEE, 2014.

[135] Matthias Schreier, Volker Willert, and Jürgen Adamy. An integrated approach to maneuver-based trajectory prediction and criticality assessment in arbitrary road environments. IEEE Transactions on Intelligent Transportation Systems, 17(10):2751–2766, 2016.

[136] Eugen Schubert, Frank Meinl, Martin Kunert, and Wolfgang Menzel. High resolution automotive radar measurements of vulnerable road users–pedestrians & cyclists. In 2015 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), pages 1–4. IEEE, 2015.

[137] Jens Schulz, Constantin Hubmann, Julian Löchner, and Darius Burschka. Multiple model unscented kalman filtering in dynamic bayesian networks for intention estimation and trajectory prediction. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 1467–1474. IEEE, 2018.

[138] Ole Schumann, Jakob Lombacher, Markus Hahn, Christian Wöhler, and Jürgen Dickmann. Scene understanding with automotive radar. IEEE Transactions on Intelligent Vehicles, 5(2):188–203, 2019.

[139] N Schüssler and KW Axhausen. Processing gps raw data without additional information, paper presented at the 88th annual meeting of the transportation research board, 2009.

[140] Linda Senigagliesi, Gianluca Ciattaglia, Adelmo De Santis, and Ennio Gambi. People walking classification using automotive radar. Electronics, 9(4):588, 2020.

[141] Milwaukee Sentinel. Phantom auto'will tour city. The Milwaukee Sentinel, page 4, 1926.

[142] Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 31, 2017.

[143] M. Sheeny, E.D. Pellegrin, Mukherjee S., A. Ahrabian, S. Wang, and A.M. Wallace. Radiate: A radar dataset for automotive perception. arXiv:2010.09076v, 2020.

[144] Marcel Sheeny, Andrew Wallace, and Sen Wang. 300 ghz radar object recognition based on deep neural networks and transfer learning. IET Radar, Sonar & Navigation, 14(10):1483–1493, 2020.

[145] Mohammad Shokrolah Shirazi and Brendan Tran Morris. Trajectory prediction of vehicles turning at intersections using deep neural networks. Machine Vision and Applications, 30(6):1097–1109, 2019.

[146] Merrill Ivan Skolnik. Introduction to radar systems. New York, 1980.

[147] Josef Steinbaeck, Christian Steger, Gerald Holweg, and Norbert Druml. Next generation radar sensors in automotive sensor fusion systems. In 2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF), pages 1–6. IEEE, 2017.

[148] Jack Stilgoe. Seeing like a tesla: How can we anticipate self-driving worlds? Glocalism: Journal of Culture, Politics and Innovation, 3(0):1–20, 2017.

[149] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. arXiv preprint arXiv:1409.3215, 2014.

[150] Kolja Thormann, Marcus Baum, and Jens Honer. Extended target tracking using gaussian processes with high-resolution automotive radar. In 2018 21st International Conference on Information Fusion (FUSION), pages 1764–1770. IEEE, 2018.

[151] T. Tieleman and G. Hinton. Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. Tech. Rep., Technical report, page 31, 2012.

[152] Rafael Toledo-Moreo and Miguel A Zamora-Izquierdo. Imm-based lane-change prediction in highways with low-cost gps/ins. IEEE Transactions on Intelligent Transportation Systems, 10(1):180–185, 2009.

[153] Quan Tran and Jonas Firl. Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression. In 2014 IEEE Intelligent Vehicles Symposium Proceedings, pages 918–923. IEEE, 2014.

[154] John R Treat, Nicholas S Tumbas, ST McDonald, David Shinar, Rex D Hume, RE Mayer, Rickey L Stansifer, NJ Castellan, et al. Tri-level study of the causes of traffic accidents. volume 1, causal factor tabulations and assessments. Technical report, United States. National Highway Traffic Safety Administration, 1977.

[155] John R Treat, NS Tumbas, ST McDonald, D Shinar, Rex D Hume, RE Mayer, RL Stansifer, and NJ Castellan. Tri-level study of the causes of traffic accidents: final report. executive summary. Technical report, Indiana University, Bloomington, Institute for Research in Public Safety, 1979.

[156] Shiu Hang Tsang, Peter S Hall, Edward G Hoare, and Nigel J Clarke. Advance path measurement for automotive radar applications. IEEE Transactions on intelligent transportation systems, 7(3):273–281, 2006.

[157] Shiu Hang Tsang, Edward G Hoare, Peter S. Hall, and Nigel J Clarke. Automotive radar image processing to predict vehicle trajectory. In Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348), volume 3, pages 867–870. IEEE, 1999.

[158] Wei-Ting Tseng, Min-Te Sun, Kazuya Sakai, and Wenlu Wang. Turn prediction for special intersections and its case study. In Proceedings of the 48th International Conference on Parallel Processing: Workshops, pages 1–9, 2019.

[159] Udacity. Udacity self-driving car driving data.

[160] D. Vasquez, T. Fraichard, and C. Laugier. Growing hidden markov models: An incremental tool for learning and predicting human and vehicle motion. The International Journal of Robotics Research, 28(11-12):1486–1506, 2009.

[161] Harini Veeraraghavan, Nikos Papanikolopoulos, and Paul Schrater. Deterministic sampling-based switching kalman filtering for vehicle tracking. In 2006 IEEE Intelligent Transportation Systems Conference, pages 1340–1345. IEEE, 2006.

[162] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence-video to text. In Proceedings of the IEEE international conference on computer vision, pages 4534–4542, 2015.

[163] R Vinayakumar, Mamoun Alazab, Sriram Srinivasan, Quoc-Viet Pham, Soman Kotti Padannayil, and K Simran. A visualized botnet detection system based deep learning for the internet of things networks of smart cities. IEEE Transactions on Industry Applications, 56(4):4436–4456, 2020.

[164] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. arXiv preprint arXiv:1511.06391, 2015.

[165] Steve Viscelli. Driverless? autonomous trucks and the future of the american trucker. 2018.

[166] Christian Waldschmidt and Holger Meinel. Future trends and directions in radar concerning the application for autonomous driving. In 2014 11th European Radar Conference, pages 416–419. IEEE, 2014.

[167] A.M. Wallace, S Mukherjee, B. Toh, and A. Ahrabian. Combining automotive radar and lidar for surface detection in adverse conditions. IET Radar, Sonar & Navigation, 15, 2021.

[168] Peng Wang, Xinyu Huang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The apolloscape open dataset for autonomous driving and its application. IEEE transactions on pattern analysis and machine intelligence, 2019.

[169] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. Annals of Data Science, pages 1–26, 2020.

[170] M. Werling, J. Ziegler, S. Kammel, and S. Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 987–993. IEEE, 2010.

[171] World Health Organization (WHO). Road traffic injuries. Road traffic injuries, —-.

[172] Wikipedia, The Free Encyclopedia. "FMCW radar diagram". https://commons.wikimedia.org/wiki/File:Bsp2_CW-Radar.EN.png. [Online; accessed June 17, 2020].

[173] Wikipedia, The Free Encyclopedia. "Synthetic-aperture radar". https://en.wikipedia.org/wiki/Synthetic-aperture_radar. [Online; accessed June 17, 2020].

[174] Hanwool Woo, Yonghoon Ji, Hitoshi Kono, Yusuke Tamura, Yasuhide Kuroda, Takashi Sugano, Yasunori Yamamoto, Atsushi Yamashita, and Hajime Asama. Lane-change detection based on vehicle-trajectory prediction. IEEE Robotics and Automation Letters, 2(2):1109–1116, 2017.

[175] Guo Xie, Anqi Shangguan, Rong Fei, Wenjiang Ji, Weigang Ma, and Xinhong Hei. Motion trajectory prediction based on a cnn-lstm sequential model. Science China Information Sciences, 63(11):1–21, 2020.

[176] 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 net-

works. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 1441–1446. IEEE, 2018.

[177] Liu Xin, Hai-Long Pei, and Li Jianqiang. Trajectory prediction based on particle filter application in mobile robot system. pages 389 – 393, 08 2008.

[178] Y. Xing, C. Lv, H. Wang, H. Wang, Y. Ai, D. Cao, E. Velenis, and F. Y. Wang. Driver lane change intention inference for intelligent vehicles: Framework, survey, and challenges. IEEE Transactions on Vehicular Technology, 68(5):4377–4390, 2019.

[179] S. Xingjian, Z. Chen, H. Wang, D. Y. Yeungn, W. K. Wong, and W. C. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In Advances in neural information processing systems, pages 802–810, 2015.

[180] Guoqing Xu, Li Liu, Yongsheng Ou, and Zhangjun Song. Dynamic modeling of driver control strategy of lane-change behavior and trajectory planning for collision prediction. IEEE Transactions on Intelligent Transportation Systems, 13(3):1138–1155, 2012.

[181] Zhanhong Yan, Kaiming Yang, Zheng Wang, Bo Yang, Tsutomu Kaizuka, and Kimihiko Nakano. Time to lane change and completion prediction based on gated recurrent unit network. In 2019 IEEE Intelligent Vehicles Symposium (IV), pages 102–107. IEEE, 2019.

[182] Lu Yang and Weiyao Lan. On secondary development of ptv-vissim for traffic optimization. In 2018 13th International Conference on Computer Science Education (ICCSE), pages 1–5, 2018.

[183] Wen Yao, Huijing Zhao, Philippe Bonnifait, and Hongbin Zha. Lane change trajectory prediction by using recorded human driving data. In 2013 IEEE Intelligent Vehicles Symposium (IV), pages 430–436. IEEE, 2013.

[184] Sung Gu Yi, Chang Mook Kang, Seung-Hi Lee, and Chung Choo Chung. Vehicle trajectory prediction for adaptive cruise control. In 2015 IEEE Intelligent Vehicles Symposium (IV), pages 59–64. IEEE, 2015.

[185] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2636–2645, 2020.

[186] Giancarlo Zaccone. Getting started with TensorFlow. Packt Publishing Ltd, 2016.

[187] Ruifeng Zhang, Libo Cao, Shan Bao, and Jianjie Tan. A method for connected vehicle trajectory prediction and collision warning algorithm based on v2v communication. International Journal of Crashworthiness, 22(1):15–25, 2017.

[188] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 12126–12134, 2019.

[189] A. Zyner, S. Worrall, and E. Nebot. Naturalistic driver intention and path prediction using recurrent neural networks. IEEE Transactions on Intelligent Transportation Systems, 2019.

[190] Alex Zyner, Stewart Worrall, and Eduardo Nebot. A recurrent neural network solution for predicting driver intention at unsignalized intersections. IEEE Robotics and Automation Letters, 3(3):1759–1764, 2018.