Vehicle Trajectory Prediction Using Road Structure

Master of Science Thesis

Geetank Raipuria

TUDelft

Vehicle Trajectory Prediction Using Road Structure

by

Geetank Raipuria

to obtain the degree of Master of Science
at the Delft University of Technology.

December 20, 2017

Student number: 4502736
Project duration: May 2017 – December 2017
Thesis committee: Prof. dr. ir. Pieter P. Jonker
Dr. Julian F.P. Kooij
Dr. Manuel Mazo Espinosa
Ir. Floris Gaisser

Faculty of Mechanical, Maritime and Materials Engineering (3mE)

TU Delft

The work in this thesis was supported by Robot Care Systems B.V., in association with the WEpods project

Copyright © Department of Cognitive Robotics
All rights reserved.

Abstract

An autonomous vehicle should be able to operate amidst numerous other human-driven vehicles, each driving on its own trajectory. To safely navigate in such a dynamic environment, the autonomous vehicle should be able to predict trajectories of the vehicles operating in its vicinity and use these to plan its own path. Most related work uses a vehicle's past trajectory to model its behavior, based on which the future trajectory is predicted. However, they do not focus on the influence of contextual features such as road structure from the scene that may affect the vehicle's future trajectory. This work proposes an approach to predict a long-term vehicle trajectory using not only the past trajectory of a vehicle but also contextual features from the driving scene. We model the road structure to help prediction on curved road sections. A Recurrent Neural Network is used to learn vehicle behavior from past vehicle trajectories and predict future trajectories while incorporating road structure. Using a trajectory dataset collected from a test vehicle, we compare our model's performance with the conventional prediction approach based on only past vehicle trajectory.

Table of Contents

Acknowledgements

This thesis represents an important milestone in my career. It has been a period of intense learning for me, not only in the scientific arena but also on the personal side.

This work was possible with the guidance of my thesis advisor, Ir. Floris Gaisser who has supported me throughout my thesis with his patience and knowledge. I attribute the level of my Master's thesis to his encouragement and inputs. His guidance helped me in research as well as writing of this thesis. I could not have imagined having a better advisor and mentor for my study.

I would like to thank Prof. dr ir Pieter P. Jonker for giving me the opportunity to work on this thesis and for his extensive feedback for successfully completing my dissertation. I am grateful to him for his valuable inputs.

I would like to thank other members of the thesis committee for help shaping the research goals and evaluating this work.

Special thanks to all the colleagues at Robot Care Systems, The Hague, for the stimulating discussions and providing a conducive work environment for me.

Importantly, I would also like to thank my parents for their wise counsel and supporting me throughout all my studies. This accomplishment would not have been possible without them.

Delft, University of Technology — Geetank Raipuria
December 20, 2017

“You can’t cross the sea merely by standing and staring at the water.”
— *Rabindranath Tagore(1861 - 1941)*

Chapter 1

Introduction

Many traffic accidents occur every year leading to injury, loss of life and property. In 2014 alone there were about a million road accidents causing 1.3 million injuries and 26,000 fatalities in Europe itself [4]. This includes vehicle to vehicle accidents and vehicle accidents with other road users like pedestrians and cyclists. The majority of these accidents involve human error caused by inattentiveness (due to tiredness, distraction or other reasons), misjudgment (decision error), skill based error, observation error due to traffic occlusion as well as poor lighting conditions and non observance of traffic laws [5, 6]. To reduce the number of accidents, the dependence on human input can be limited by increasing the degree of automation in the driving task. Many corporations as well as research institutions, have been working on improving vehicle safety through added intelligence in vehicles.

The aim to reduce vehicular traffic accidents has led to a tremendous research in intelligent vehicles, with the goal to make transportation completely autonomous. Driver Assistant systems such as Adaptive Cruise Control, Collision Detection and Braking Systems as well as Lane Change Assistance already automate the driving task especially on highway roads. To fully automate vehicles, it is essential is to enable safe vehicle operation in complex scenarios without the input from the human driver. This requires path planning through the vehicles environment.

The environment of an autonomous vehicle contains numerous human driven vehicles which act as dynamic obstacles. Potentially, the trajectory of each of these vehicles could intersect with the planned trajectory of the ego-vehicle. That is, each vehicle in the vicinity could be moving into a collision state with the autonomous vehicle. To ensure safety and prevent planning its path into a collision, the autonomous vehicle needs to know the future trajectories of vehicles in its vicinity. However, each of these dynamic obstacles follow their own planned trajectories which are unknown to the autonomous vehicle. As a result, the vehicle needs to predict the future trajectories of the vehicles operating in its vicinity.

The trajectory of a vehicle is the result of two contributing factors: First the behavior of a vehicle's driver, such as changing a lane that reflects maneuver intention. Secondly, external factors such as road structure which affect the trajectory of the vehicle during driving in that

maneuver. Both can be inferred from clues or 'indicators' obtained from the vehicle states and the driving scene.

We categorize these indicators into four types: Motion Indicators, Infrastructure Indicators, Object Indicators and Interaction Indicators. Motion Indicators consists of position, velocity, acceleration and orientation of a dynamic object over past and current time frames. These features can be used to recognize initiation of a maneuver such as acceleration or lane change. Object indicators communicate the intention of a vehicle to execute a maneuver such as turn and brake signals. Both Motion and

Object Indicators together capture behavior of an object. Infrastructure Indicators comprise features in the road infrastructure that regulate the flow of traffic. This primarily includes road geometry, pedestrian crossing, traffic signals and other traffic regulatory features. Lastly, the Interaction Indicators includes the interaction between traffic participants i.e. vehicle to vehicle or vehicle to pedestrian interactions. Infrastructure and Interaction indicators together capture features which affect the behavior of the object.

A widely used approach for trajectory prediction is based on a vehicle's past kinematic states[7], i.e. only Motion Indicators. Such an approach does not consider the influence of external factors like road structure or interaction with other traffic participants that may cause change in a vehicle's trajectory. As a result, prediction based on only Motion Indicators may not perform well in complex driving scenarios and is limited to short-term prediction [7]. Furthermore, another set of literature focuses on identifying vehicle maneuver intention. For example, turning on an intersection [8, 9] or lane change [10, 11]. Trajectory prediction is then done based on identified intention [12, 7]. These approaches use indicators like geometry of the road, speed limit and driving style for identifying a vehicle's intention. However, they do not consider these indicators during the vehicle trajectory prediction.

The aim of this work is to consider Motion and Infrastructure Indicators in trajectory prediction. Within the scope of this work, we only consider vehicles driving on a stretch of road such that vehicles are unconstrained by other vehicles and intersections of the road.

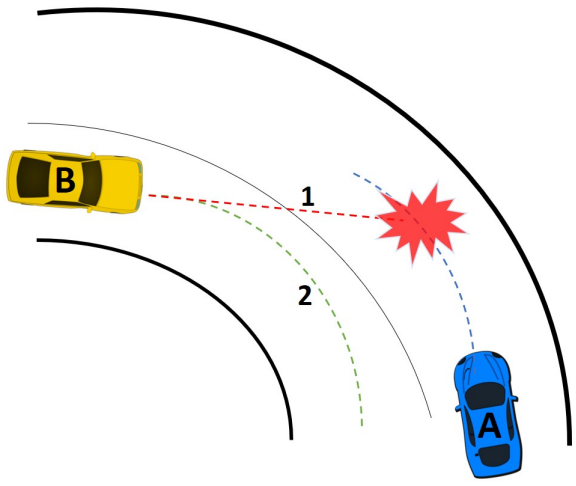


Figure 1-1: Vehicle trajectory prediction on a curved road section

A good example of a scenario this work addresses is shown in figure 1-1. The ego-vehicle (A) is approaching a curve. On the opposite side of the road, another vehicle (B) is approaching from the other direction. A prediction approach based only on past trajectory of vehicle B would predict a path (1), which would collide with the intended path of the ego-vehicle. This predicted path would call for an unnecessary evasive maneuver, since it can be expected that vehicle B would follow path (2) which follows the road.

The above scenario is one of the motivations to focus on improving vehicle trajectory prediction on curved road sections. In related work [18, 12], only limited attention is given to predict vehicle trajectories considering the influence of road structure. Therefore, this work will investigate how to incorporate the road structure into the modeling of the vehicle as well as on how to model the vehicle.

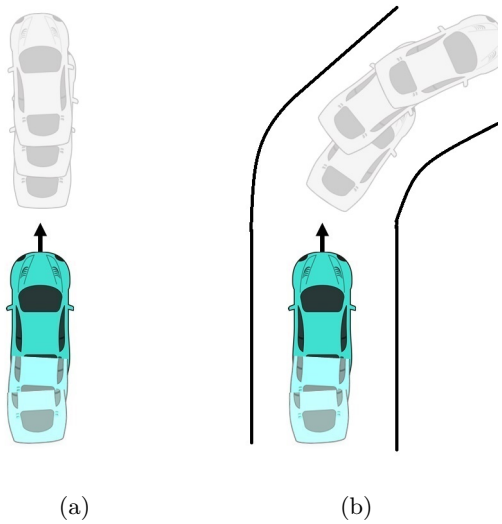
The main contributions of this work are: First, the development of features for using Road Structure as a context in trajectory prediction on curved road sections. Second, presenting a modeling approach which can exploit the variety of indicators in addition to Motion Indicators for reliable long term trajectory prediction.

The thesis is organized as follows: Chapter 2 describes our approach to model the road structure such that they can be it for trajectory prediction. Chapter 3 presents two models which use the Motion and Infrastructure Indicator for trajectory prediction. These contributions are evaluated experimentally in Chapter 4 and discussed in Chapter 5.

Chapter 2

Road Structure as Infrastructure Indicators

Consider a vehicle driving in the vicinity of an autonomous vehicle. To assure that the two vehicles do not collide, the autonomous vehicle needs to predict the future trajectory of this vehicle, while planning its own path. A prediction can be made using the vehicle's kinematic states, i.e. its position, orientation and velocity, as seen in figure 2-1(a). Based on this information, the vehicle would be predicted to move straight along the current direction of the vehicle velocity. However, if the road makes a turn, as seen in figure 2-1(b), the predicted path is not correct. This is because, while making the prediction, no information about the road structure was used.



(a) (b)

Figure 2-1: Trajectory Prediction using (a) only vehicle motion history, (b) vehicle motion history along with road structure

Vehicular traffic is constraint by an elementary traffic, to drive only on the predefined roadway. For example, a vehicle driving on a curved road would move roughly along the road curvature, to prevent going off-road. Rarely do vehicles leave this drivable roadway, except to halt or under certain exceptional circumstances. It is reasonable to assume that vehicular traffic will move along the predefined roadway path. This constraints vehicle motion by the structure of the roadway.

A better path prediction can be made considering information about the structure of the roadway. With this improved prediction ability, vehicle trajectories can be predicted more accurately over a longer duration of time, with the assumption that vehicles will roughly follow the road structure. The further in time an autonomous vehicle can reliably predict the motion of a vehicle operating in its environment, the more time it would have to plan its path. This is an important step towards safer autonomous driving.

This chapter describes how the influence of road structure on a vehicle trajectory can be modeled, such that it can be incorporated into path prediction.

2-1 Curvilinear Coordinate System

Consider a vehicle moving on a curved path and assume that vehicles only move on the roadway. The vehicle's state would change along the geometry of road as the vehicle drives forward. That is, the vehicle future states are constrained along the road structure, which is called the Road Geometry Constraint [1]. However, the motion state in the Global Cartesian Coordinate Systems (GCCS) changes freely along the Global X and Global Y axis. That is, the Global Cartesian Coordinate Systems does not constraint the vehicle motion along the road geometry.

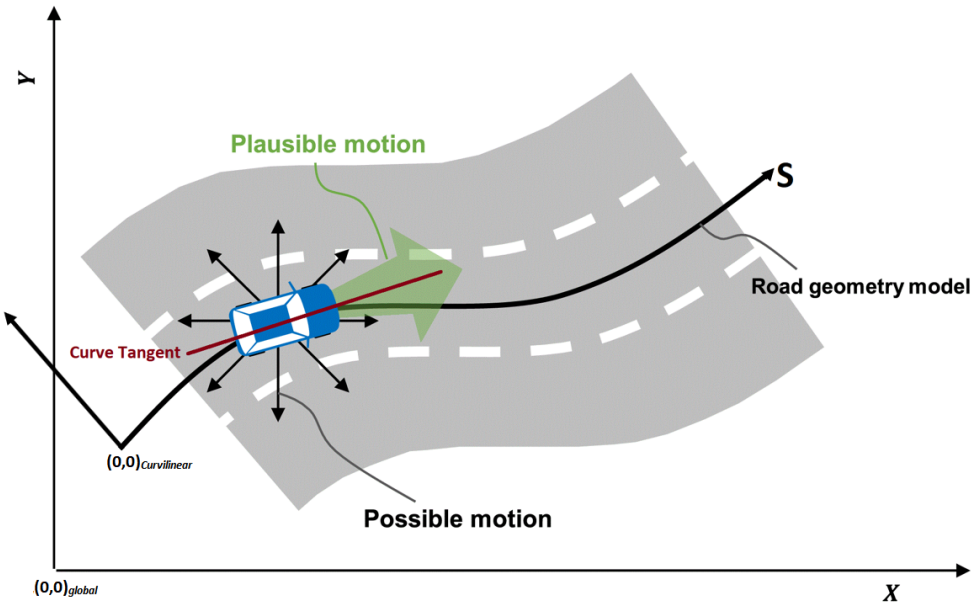


Figure 2-2: The Curvilinear Coordinate System, image adopted from [1]

To incorporate this constraint, the vehicle motion can be described as a combination of motion on an axis which runs along the roadway geometry (axis S) and an orthogonal axis (axis N). The motion on axis S represents longitudinal vehicle motion, whereas motion on axis N represents motion in the lateral direction. As a result, the motion description becomes a function of the road geometry.

The two axes form an orthogonal space and introduce a new coordinate system, a Curvilinear Coordinate System (CCS) [1], as shown in figure 2-2. In this coordinate system the driving direction of the road is always aligned with the axis S of the coordinate system.

The significance of CCS can be seen from another perspective. Vehicles are non-holonomic in nature and can maneuver only in certain directions. Not considering this causes the inclusion of more degrees of freedom than the vehicle actually possesses, thereby increasing the complexity of path prediction. Furthermore, the most probable motion of a vehicle is constrained by the geometry of the road and the direction of the traffic flow on the road. As can be seen in figure 2-2, a vehicle moving forward along the roadway would only maneuver in limited directions, which is defined by the tangent to the roadway curve. Describing vehicle motion in CCS provides this constraint by making the vehicle's motion a function of the road geometry. Thus allowing to infer the most probable direction of vehicle's motion.

The CCS is used by [1] to improve tracking and classifying vehicle behavior using the roadway geometry information. Specifically, CCS is used to differentiate between lateral vehicle motion due to a lane change or lateral motion to follow a curved road. We further extend the use of the CCS to incorporate the roadway geometry information in trajectory prediction. We use the assumption that a vehicle would follow the curvature to predict the vehicle's trajectory along the road geometry.

2-1-1 Defining CCS

The Curvilinear Coordinate System is based on roadway geometry, and requires its representation in the form of a smooth continuous curve which forms the longitudinal axis S. The distance on this *roadway curve* from the point of origin would be the x coordinate in CCS. [1] uses a B-spline to define the roadway curve. We use a similar approach described in the following paragraphs.

Given the complexity of the roadway geometry which is often consisting of multiple curved paths of varying curvature and length, the roadway path is broken into smaller sections. Each section has a starting node and an end node (control points) which are shared with adjoining curves, see figure 2-3(a). On each of these sections a spline curve is fit which is represented by the following parametric equation:

$$X = a_x * s^3 + b_x * s^2 + c_x * s + d_x \tag{2-1}$$

$$Y = a_y * s^3 + b_y * s^2 + c_y * s + d_y \tag{2-2}$$

where X and Y are the coordinates in GCCS, s is the parameter which varies from 0 to a value k for each polynomial curve, $a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y$ are constants. This gives a piecewise

polynomial curve, which is continuous in the 1st (velocity) and 2nd derivative (acceleration). The lateral axis n is a perpendicular to this piecewise polynomial curve at every point on the curve. The derivative of the curve is used to find the perpendicular axis.

Extending the CCS definition of [1], we define a separate curve for either direction of traffic on the road, as seen in figure 2-3(a). This is due to two reasons. A vehicle would move only in the right direction of traffic, i.e. the vehicle is not only constrained by its non-holonomic motion but also additionally by the direction of traffic flow. Defining a single curve for the entire roadway, would not allow differentiating between vehicles traveling in either direction of the road. To provide this directional constraint, separate curves are fit on both sides of the road. Thus, a vehicle's motion is constrained to the positive direction of its curve tangent.

Furthermore, in certain situations, the road curvature may be different for either side of the road. This can be visualized in figure 2-3(b). The radius of the curve varies for either direction of the traffic flow, which changes the distance traveled along the curve. As a result fitting a single curve for both sides of traffic flow would give an inaccurate description of the road structure.

2-1-2 Using Curvilinear Coordinate System in Path Prediction

To incorporate the road geometry in path prediction, the vehicle trajectory needs to be transformed from GCCS to CCS. After prediction, to use in further path planning, the predicted trajectory can be converted back to the GCCS. This section describes the process to transform a vehicle state from GCCS to CCS and back.

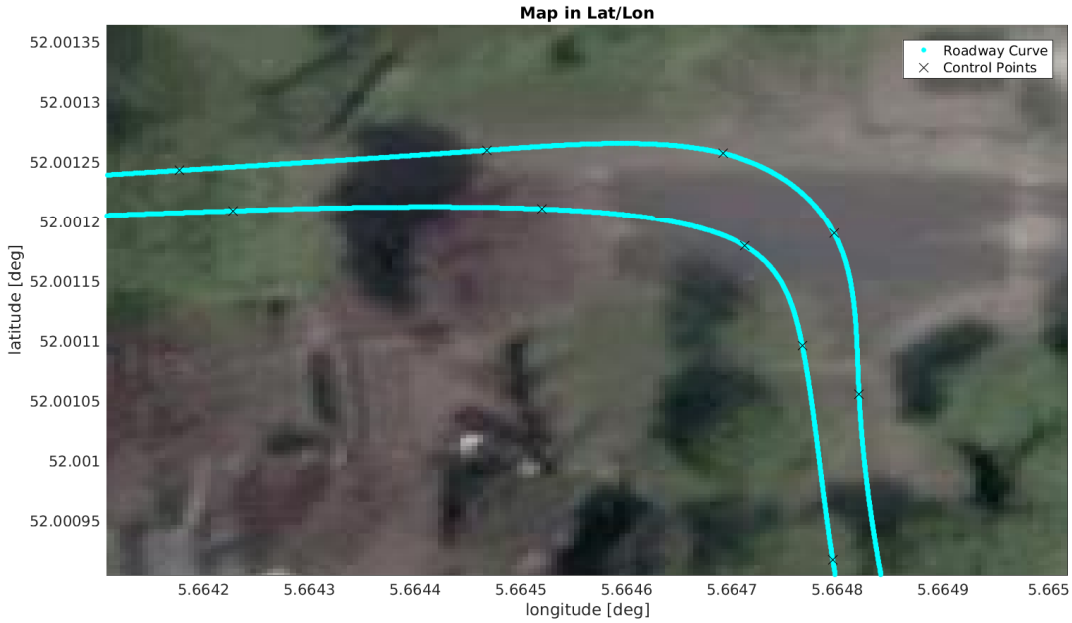
A vehicle state in the Curvilinear Coordinate system (CCS) is defined as $(X^C, Y^C, VX^C, VY^C, \phi^C)$, where X^C and Y^C are the position of the vehicle along the longitudinal axis S and the lateral axis N. VX^C, VY^C are corresponding velocities and ϕ^C is the vehicle's orientation in the CCS. Similarly $(X^G, Y^G, VX^G, VY^G, \phi^G)$ are vehicles state in GCCS.

To aid the transformation of vehicle states between the two coordinate systems, we define an intermediate Local Cartesian Coordinate System (LCCS), similar to GCCS. The origin of LCCS is at the point c_p, such that c_p is the closest point on the roadway curve to vehicle position in GCCS (X^G, Y^G) on the roadway curve. Point c_p is represented by (X_p^G, Y_p^G) which is a function of value s in the parametric equation 2-1 and (X_p^C, Y_p^C) in CCS, where Y_p^C is equal to zero as the point lies on the longitudinal axis S itself. The X axis of this coordinate system is aligned with the tangent to the roadway curve and the Y axis runs perpendicular to the tangent. This is different from CCS in two ways: The origin of CCS is the starting point of the roadway curve rather than point c_p. the longitudinal axis of LCCS is the tangent of the curve, whereas for CCS it is the roadway curve itself.

Transforming from GCCS to CCS

Given coordinates of a vehicle in GCCS $(X^G, Y^G, VX^G, VY^G, \phi^G)$, X^C, can be obtained by finding the closest point c_p on the roadway curve and the distance from the curve origin to point c_p along the curve. As shown in figure 2-4. Y^C, is obtained by calculating the distance of vehicle position (in GCCS) from the point c_p along the Y axis of LCCS which is

(a)



(b)

Figure 2-3: a) Roadway Curve plotted on GPS map, b) Figure showing different Roadway Curve for either side of the road

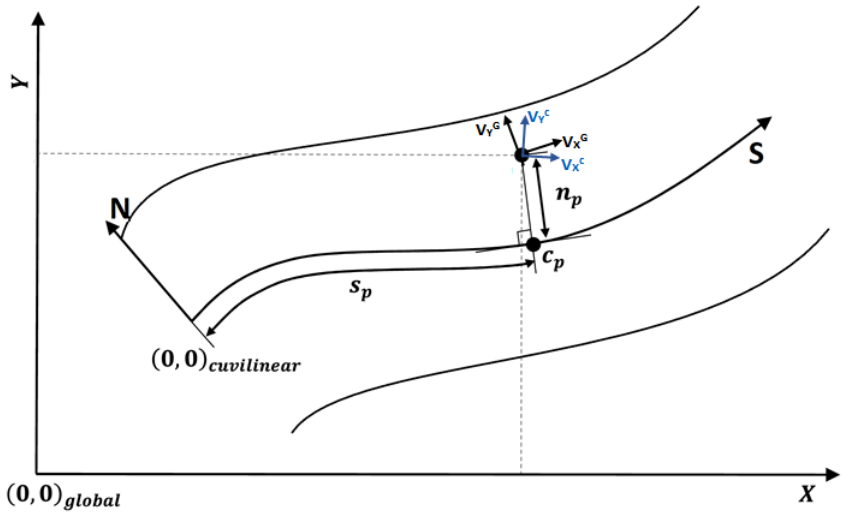


Figure 2-4: The Curvilinear Coordinate system, image adopted from [1]

perpendicular to the roadway curve tangent. Velocity VX^C and VY^C in the CCS are the velocities of the vehicle along the tangent to the parametric curve and perpendicular to the tangent direction. In the next paragraphs this process is described in more detail.

Obtaining X Coordinate in CCS - X^C
This includes finding the closest point c_p to (X^G, Y^G) on the roadway curve represented by s , and obtaining the distance along the curve from the curve origin to the point c_p.

To find point c_p, the following procedure is used:

- The two nodes of the roadway curve closest to X^G, Y^G are found,
  this allows selection of the corresponding section C of the roadway curve that contains the point c_p. As we define a curve for either side of the road, it is essential that only the correct set of control points are chosen. Traffic flow on either side of a road would have an orientation closely aligned with the tangent to the curve as it follows the road curvature. This along with the vehicle's orientation ϕ^G is used to select the correct set of nodes.
- Next, using the parametric equations representing the curve section C obtained from previous steps, closest point c_p on the curve to the vehicle position X^G, Y^G is found. This is achieved by using a non-linear optimization to minimize a function f.

  $$f = (a_x * s^3 + b_x * s^2 + c_x * s + d_x - X^G)^2 + (a_y * s^3 + b_y * s^2 + c_y * s + d_y - Y^G)^2 \quad (2\text{-}3)$$

  where s is the unknown parameter, $a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y$ are constants representing the parametric curve. The optimization procedure minimizes the distance between the point (X^G, Y^G) and an unknown point c_p (X_p^C, Y_p^C) on the curve, solving for a s value (see equation 2-1). This provides the location of point c_p.

The next step is to obtain the distance of the point c_p from the curve origin. For this, we add the lengths of each section of the roadway curve until the section C and the length of

the section C until point c_p. Each section is defined by a set of parametric equations 2-1, and arch length for each parametric curve section can be obtained as [13]:

$$L = \int_0^{s=k} dl \tag{2-4}$$

where

$$dl = \sqrt{1 + \left(\frac{dy}{dx}\right)^2} dx \tag{2-5}$$

This can be further resolved as:

$$\begin{aligned} dl &= \sqrt{1 + \left(\frac{dy}{dx}\right)^2} dx \\ &= \sqrt{1 + \frac{\left(\frac{dy}{dt}\right)^2}{\left(\frac{dx}{dt}\right)^2}} \frac{dx}{dt} dt \\ &= \frac{1}{\left|\frac{dx}{dt}\right|} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} \frac{dx}{dt} dt \\ &= \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} dt \end{aligned} \tag{2-6}$$

This gives the length of a parametric curve as:

$$L = \int_0^{s=k} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} dt \tag{2-7}$$

Thus the distance from the curve origin to point c_p is given by

$$l = \Sigma_{i=1}^{C-1} L_i + L_c \tag{2-8}$$

where L_i is the length of curve i, and L_c is the length of curve C until point c_p.

Obtaining the Y Coordinate in CCS - Y^C
Y^C represents the coordinate on a perpendicular axis to the tangent at point c_p. Essentially Y^C is the signed distance between point c_p and (X^G, Y^G) in GCCS. To obtain Y^C, the tangent to the curve at c_P is found, followed by obtaining the distance of (X^G, Y^G) from point c_P along a perpendicular to the curve tangent. The procedure is detailed as follows:

- The tangent at point c_p is defined by the slope of the parametric curve at that point. The slope is found using the following equation:

$$slope = \frac{dy}{dt}\frac{dt}{dx} \tag{2-9}$$

- To obtain Y^C, not only the distance between point c_p and (X^G, Y^G) is required, but also the direction. Rather than merely taking the numerical distance, the LCCS is used. (X^G, Y^G) is transformed to the LCCS (X^L, Y^L) using a translation and rotation operation given by the equation .

$$\begin{bmatrix} X^C \\ Y^C \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} * \begin{bmatrix} X^G - X_p^G \\ Y^G - Y_p^G \end{bmatrix} \tag{2-10}$$

  Where θ is the slope of the tangent at point c_p. Now, the Y coordinate of the object location in this local coordinate system Y^L is also the Y^C.

Obtaining Velocity in Curvilinear Coordinate System
The velocity of the object VX^C and VY^C in CCS, are velocity components along the tangent to the curve at point c_p and perpendicular to tangent. To obtain these components, VX^G and VY^G are transformed to the local Coordinate system using a 2D rotation matrix, given by the equation

$$\begin{bmatrix} VX^C \\ VY^C \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} * \begin{bmatrix} VX^G \\ VY^G \end{bmatrix} \tag{2-11}$$

Where θ is the slope of the tangent at point c_p. This velocity vector represents the velocity of the object in CCS.

Transforming from CCS to GCCS

A vehicle position in CCS is the distance along the roadway curve and a distance perpendicular to this curve. To find the position of a vehicle in GCCS from CCS, the first step is to find the curve section C of the piecewise polynomial road curve such that a closest point c_p lies on that curve section. Once the parametric curve is obtained, a corresponding value s can be determined which represents the location of the point c_p on the curve. This value s gives gives the X^G using the parametric equation 2-1. Next, using the slope of the curve at c_p, a point at distance Y_C from point c_p along the perpendicular line can be obtained. To find the vehicle's velocity in GCCS, $V^C X$ and VY^C can be multiplied with the inverse of the rotation matrix given by equation 2-11. The procedure is detailed in the following paragraphs:

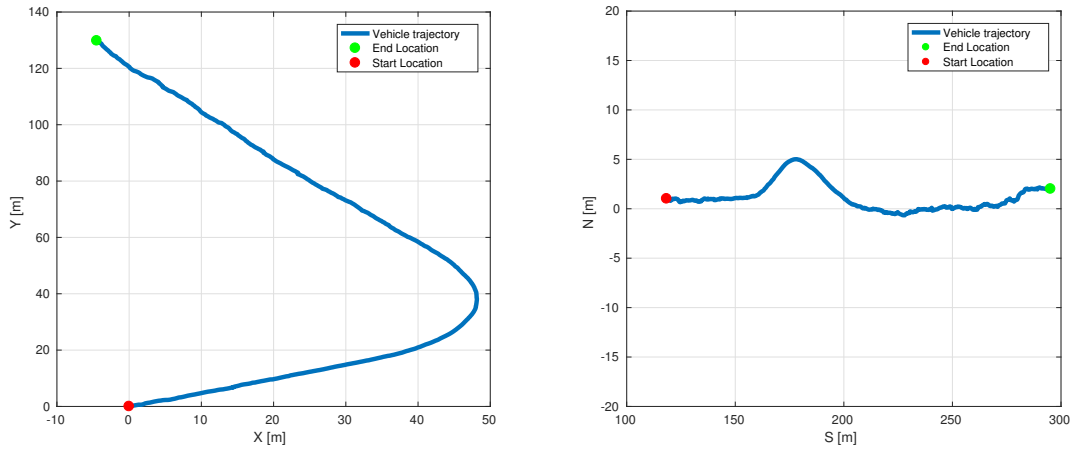
- X^C represents the distance of the closest point c_p on the CCS X axis from the roadway curve's origin, as defined in the previous section. To obtain c_p, we use equation 2-8. For faster computation, a look-up table is prepared with cumulative lengths for parametric curves starting from the road origin. This directly provides the parametric curve C that contains point c_p. The obtained value s from equation 2-8 can be plugged in the parametric curve equation 2-1 to give the location of point c_p in the GCCS. The X coordinate of point c_p is also the X coordinate of the vehicle location in GCCS.
- To find Y^G, the process used to find Y^C(given by equation 2-10) is inversed, ie. the Y coordinate is first rotated back about the point c_p followed by a translation.
- The velocities are obtained by multiplying the velocity vector in CCS with the inverse of the rotation matrix given in equation 2-11.

Conclusion

To incorporate the roadway geometry information into the path prediction, a vehicle trajectory described in GCCS needs to be converted into CCS. This conversion makes the trajectory a function of the road geometry curve and allows the prediction of future trajectories along this curve. The prediction model can predict vehicle trajectory in this transformed motion states. A prediction model can predict vehicle trajectories in CCS. The predicted trajectory can then be transformed back to GCCS after prediction in CCS. The predicted trajectory should be much more similar to the real vehicle trajectory, than predicting in GCCS.

2-1-3 Effect of CCS on vehicle motion description

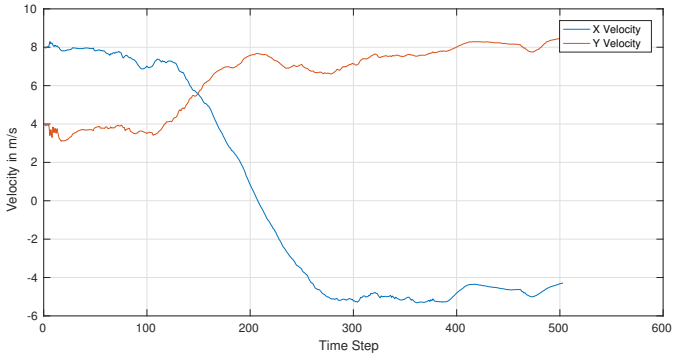
Figure 2-5 and 2-6 show the motion of a vehicle in both GCCS and CCS. Observing vehicle motion in GCCS (fig 2-5(a)), it is difficult to draw any conclusion on the vehicle's driving behavior. For example, is the vehicle following its lane or is it also moving laterally (lane change)? This is because there is no information available about the road structure. However, in CCS the axis S is aligned with the road geometry, and axis N is perpendicular to axis S; which inherently provides information about the road structure. This allows distinguishing between vehicle longitudinal motion (forward vehicle motion along the road geometry) and lateral motion (vehicle motion away from or towards the lane center). That is, the vehicle's motion is now seen from the vehicle perspective itself. Figure 2-5(b), shows the corresponding trajectory in CCS.



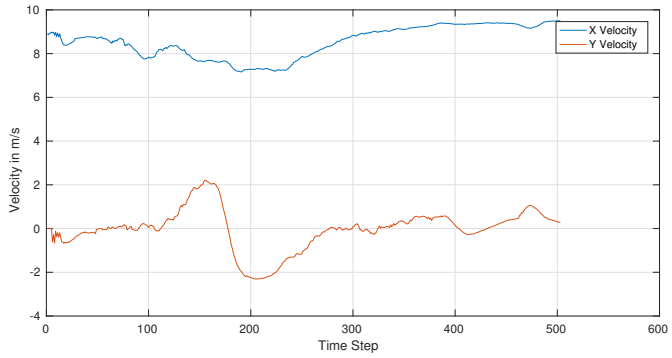
(a) (b)

Figure 2-5: Vehicle trajectory plotted (a) in GCCS (b) in CCS

For a vehicle traveling along the road, its longitudinal motion would be much larger than its lateral motion, as seen in figure 2-5(b) and 2-6(b). Describing the vehicle's motion in CCS allows predicting the two motion types independently, which was not possible in GCCS. That is, CCS allows to predict a vehicle to move with varying longitudinal velocity without directly affecting the lateral motion. A vehicle with no lateral motion is thus represented by a 1D



(a)



(b)

Figure 2-6: Vehicle velocity plotted (a) in GCCS (b) in CCS

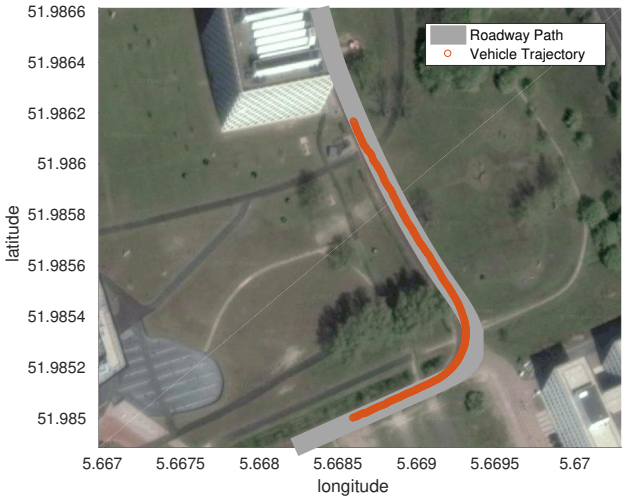


Figure 2-7: Example vehicle trajectory plotted on GPS map, along with the roadway curve

motion trajectory. This provides a powerful predictive ability. Furthermore, since we define a separate curve for either driving direction on a road, a vehicle would always travel in the positive direction of the S axis. This assumes that a vehicle never goes in the opposite traffic direction.

Figure 2-6(b) shows the vehicle's velocity in CCS. The velocity along the N axis varies on either side of the axis with a near zero mean. Since the S axis is aligned with the road geometry, such motion can easily be interpreted as lane following. Furthermore, distinguishing between lateral and longitudinal motion allows to interpret behaviors like oscillation in the lateral position (between 100 and 250 time step) as the vehicle negotiates the curve, which was not possible directly from motion description in GCCS.

2-1-4 Conclusion

A vehicle is constrained by the road structure, which affects its motion. To predict the vehicle's future trajectory it is thus essential to consider the structure of the road a vehicle drives upon. For this, we need to model the road structure such that it can be used by the prediction model.

A curvilinear coordinate system based on the road geometry is used to incorporate a constraint on the vehicle's motion due to its non-holonomy and direction of traffic flow. The longitudinal axis of this coordinate system aligns with the road geometry. This allows differentiating between longitudinal motion along the road and lateral motion, that provides powerful inferencing ability to model and predict vehicle motion.

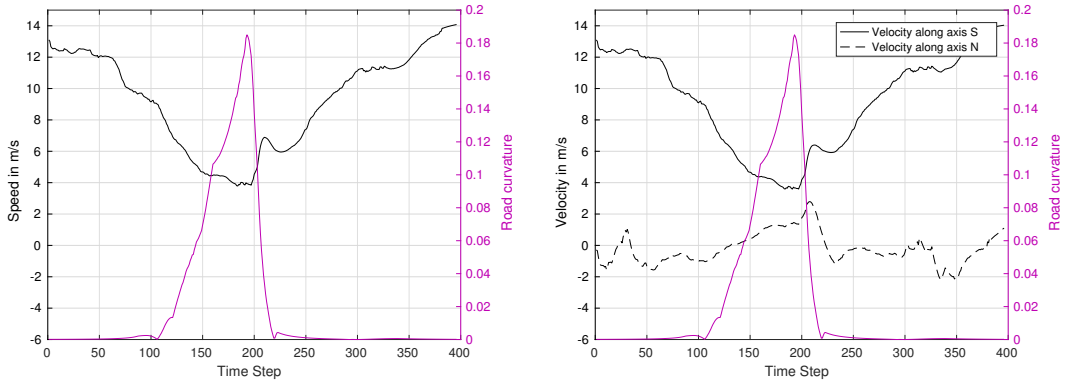
However, there is an important shortcoming of CCS. Consider a vehicle moving on a curved path and another vehicle on a straight road. In CCS both motions would be described on an axis along the road curvature. However, aligning of the longitudinal axis with the road structure removes any information about the curvature of the road itself. Both the vehicle motions appear essentially the same, and it is not possible to differentiate between a curved road and straight road.

2-2 Road Curvature

Vehicles typically lower their speed as they approach a curved section of the road. This reduction in speed allows vehicles to safely negotiate the curve without going off-road. Similarly, vehicles raise their speed towards the end of a curved section. The speed of a vehicle is thus influenced by the curvature of the road it is traveling on. Figure 3-3(a) show variation in speed of a vehicle as it negotiates a curved road section, plotted against road curvature.

Transforming the vehicle velocities to CCS (see figure 2-8(b)), it can be visualized that the vehicle's longitudinal velocity (along axis S) is the velocity component which causes this variation in speed. Using the road curvature information, this variation in velocity of a vehicle can be reasoned as the vehicle approaches or exits a curved section of the road. Furthermore, this can be used in trajectory prediction to anticipate the effect that road curvature would have on the future trajectory of a vehicle as it negotiates a curve.

Although CCS provides the structure of the road, it does not provide does information on the curvature of the road itself. Without this information, a prediction model will have no means



(a) (b)

Figure 2-8: Vehicle velocity variation plot alongside road curvature. (a) A vehicle exhibits sharp reduction in velocity as it traverses a very sharp curve (b) Smaller variations in road curvature have minor impact on vehicle velocity

to learn the existence of a curve and to anticipate variations in vehicle velocity. Thus, it is required that the road curvature is calculated separately, and used as an additional feature for the prediction. By considering the road curvature as context, a trajectory prediction model can anticipate the variation in velocity and make a better prediction of the vehicle's trajectory. This section defines the curvature of a curve and presents how it can be used to improve path prediction.

Curvature Definition

The curvature of a curve is defined as the rate of change of the angle of the tangent to the curve with respect to the curve length. This is denoted by the following equation:

$$slope = \frac{d\phi}{dl} \tag{2-12}$$

where ϕ is the angle of the tangent and l is the curve length. It is also the inverse of the curve radius at any position on the curve.

The curvature is essentially the rate of change of the tangent slope, however, the derivative is with respect to the roadway length. That means, if the tangent slope changes by significantly in short length of the curve, the curvature is large. Curvature represents how quickly does the curve tangent changes along a length of the curve.

Obtaining Road curvature

The roadway curve in used to obtain the road curvature. This is the same parametric curve which was also used to define the longitudinal axis of CCS.

However, we do not have a direct relation between the road tangent angle and curve length. For this we further resolve equation 2-12 as follows [14]:

$$\begin{aligned} dl &= \frac{d\phi/ds}{dl/ds} \\ &= \frac{d\phi/ds}{\sqrt{\dot{x}^2 + \dot{y}^2}} \end{aligned} \tag{2-13}$$

where x and y are obtained from the parametric curve equation 2-1, $\dot{x} = \frac{dx}{ds}$ and $\dot{y} = \frac{dy}{ds}$
Further,

$$\begin{aligned} tan(\phi) &= \frac{dy}{dx} \\ &= \frac{dy/ds}{dx/ds} \\ &= \frac{\dot{y}}{\dot{x}} \end{aligned} \tag{2-14}$$

and,

$$\begin{aligned} \frac{d}{dt}tan(\phi) &= sec^2\phi \frac{d\phi}{dt} \\ &= \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2} \end{aligned} \tag{2-15}$$

where $\ddot{x} = \frac{d^2x}{ds^2}$ and $\ddot{y} = \frac{d^2y}{ds^2}$
this gives,

$$\begin{aligned} \frac{d\phi}{dt} &= \frac{1}{sec^2\phi}\frac{d}{dt}(tan\phi) \\ &= \frac{1}{1 + tan^2\phi}\frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2} \\ &= \frac{1}{1 + \frac{\dot{y}^2}{\dot{x}^2}}\frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2} \\ &= \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2} \end{aligned} \tag{2-16}$$

Combining equation 2-17 with equation 2-13, gives:

$$k = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\left(\dot{x}^2 + \dot{y}^2\right)^{(3/2)}} \tag{2-17}$$

Using equation 2-17 and the roadway curve, a curvature value can be found for every location of the roadway.

Using Road Curvature

The roadway curve can be used to obtain road curvature of the road section on which a vehicle is traveling. Curvature information can then be used as a feature for a prediction model. This allows the model to anticipate the variation in the vehicle's velocity due to road curvature and in turn better predict the vehicle's position.

It is essential to mention here that a vehicle lowers its speed prior to actually entering a curve, with a deceleration comfortable for the passengers. The reason for this is two-fold. One, as soon as a vehicle enters the curve, the centrifugal forces start acting which can throw the vehicle off-road. To maintain control, the velocity needs to be lowered before the centrifugal forces becomes too high. Lowering velocity on entering the curve would require sudden braking which is not desirable. Secondly, braking in the curve itself would increase the forces on the passengers. Meaning that a driver does not lower the speed based on the current curvature of the road, but rather curvature of the road ahead.

Given two vehicles both approaching a curve, one moving fast and the other moving slow, the one traveling faster will need to brake earlier to reach a safe speed prior to entering the curved road section. The faster the vehicle travels, the more distance is needed to reduce the vehicle's speed. Therefore, the speed reduction is based on the curvature of the road x meters ahead, and the value x depend upon the velocity of the vehicle. A vehicle with a higher speed should observe the curvature of the road further ahead in order to lower the speed in time before reaching the curve. Thus to anticipate a vehicles velocity variation, a prediction model needs the curvature information not from the current road section, but rather few meters ahead. Using the curvature of the current section of the road as a feature, the model would always lag behind in predicting the velocity variation.

In addition to the vehicle's speed, the distance d_c from the beginning of the curve section at which a vehicle would begin to decelerate also depends upon the following factors:

- **The maximum curvature of the curve**
  The Higher the maximum curvature, the higher the vehicle 's deceleration needs to be. Given a maximum deceleration for comfortable breaking (about $2m/s^2$ [15]), a higher reduction in speed would a require longer breaking distance.
- **Frictional coefficient and Road Banking**
  The coefficient of friction between the road and the vehicle's tires μ as well as the road banking effect the maximum centrifugal force a vehicle can take as it negotiates the curve. This in turn dictates the maximum velocity that the vehicle can have on the curved road, given by the following equation:

$$v = \sqrt{\frac{(e + \mu) * g}{k}} \tag{2-18}$$

where e is the tangent function of banking angle, g is the gravitational acceleration, and k is the road curvature. The lower the maximum velocity in the curve is, the larger the braking distance required.

- **Mass of the vehicle**
  A vehicle with higher mass would require a longer distance for breaking before entering the curve. This is because a heavier vehicle has a higher momentum, which requires higher forces to reduce speed.
- **Driving style of the driver**
  Lastly, every driver may have different behavior while driving on a curved road. Some drivers may only slow down to the maximum safe driving speed possible on the curve, whereas some driver may further lower their vehicle's speed.

Within the scope of this work, we perform vehicle trajectory prediction based on vehicle kinematics and do not consider frictional coefficient, road banking and mass of the vehicle. Furthermore, we do not explicitly model the driver's driving style and influence of curvature on the distance required to break. However, this can be learned by a data-driven prediction model.

2-3 Conclusion

The road structure constraints vehicle motion as the vehicle drives along the roadway. To predict a vehicle's future trajectory, it is essential to incorporate the influence the road structure has on a vehicle's trajectory. For this, we model the road structure using a Curvilinear Coordinate System and road curvature value, which can be used by a trajectory prediction model.

CCS provides the description of the road structure by using the roadway curve as coordinate axis. This coordinate system distincts between vehicle motion along the road curvature and lateral motion. The road curvature information enables reasoning and predicting vehicle speed variations on a curved road section. A higher road curvature dictates lower vehicle speed due to centrifugal forces.

The two together provide a powerful context to help trajectory prediction. A vehicle motion in GCCS can be transformed to motion in CCS. A prediction model can use this motion description to predict the vehicle's future trajectory based on the past trajectory. As the longitudinal axis of CCS follows the road geometry, the vehicle's trajectory would be predicted as a function of the road geometry. This should immediately improve the predictive ability of the prediction model with additional information about the road geometry. Further improvement of the prediction can be brought about by using road curvature as a feature to the prediction model.

Chapter 3

Models for Path Prediction

An autonomous vehicle driving on a busy street would encounter numerous other vehicles operating around it. For safe interaction with these traffic participants, the autonomous vehicle will need to plan its path to avoid any potential collision. Therefore, it is essential that the autonomous vehicle predicts the future path of the vehicles operating around it. Human drivers make this prediction intuitively by observing the states of surrounding vehicles as well as the environment. Autonomous vehicles can predict the future path of the surrounding vehicles by using data on their past states to learn their behavior, as well as considering the factors that could affect their future path such as road structure. This would require a model that can analyze a vehicle's past trajectory and predict a likely future trajectory.

Such a model would be tasked to learn the behavior of a vehicle from its past trajectory. For example, whether a vehicle moves with roughly constant velocity or does it accelerates. To incorporate road structure in trajectory prediction, the model should be able to exploit motion description in Curvilinear Coordinate System as well as associate variations in the vehicle trajectory with road curvature and reflect this in the trajectory prediction. For example, the future trajectory of a vehicle driving on a curve can be predicted by obtaining its velocity and acceleration from the past trajectory and anticipating the slowing of the vehicle due to the road curvature as it follows the curve.

This chapter describes the modeling approaches which can use both Motion as well as other indicators for long-term vehicle path prediction.

3-1 Criteras for selecting prediction model

The following section lists and detais the criterea for selecting a prediction model. The criteria are:

- **The model can operate on Sequential data**
  For the path prediction task, the input to the model and the expected output is a

sequence of data which is temporally related. It is thus essential that a model should be able to make temporal relationships not just within the input data and output data, but between the two.

- **The model can work with variable input and output time steps**
  An autonomous vehicle would need to predict the trajectory of a vehicle as soon as the vehicle is observed. At the same time, the autonomous vehicle could have a longer motion history of the vehicles around it, which can be used to improve the predicted trajectory. To allow such functionality, the prediction model should be able to accept variable lengths of input. Also, the vehicle may need to predict for different time horizons, which the model should also allow.

- **The model can handle non-linearities**
  The motion of vehicle can be non-linear as it maneuvers along the road, for example acceleration is quadratic on position. To predict such motion, the models should be able to model non-linear behavior.

- **The model can operate on kinematic as well as non-kinematic features**
  The model should not be limited to predicting trajectories only based on the past kinematic states (position, velocity and acceleration). Rather, the model should also be able to associate variation in vehicle states with non-kinematic features such as road curvature and use these to improve prediction performance.

- **The model can be easily extended to incorporate new features**
  The model should be able to add a new (dimension) feature to the input data to allow flexibility and easy adaptation to predict trajectories in more complex scenarios.

3-2 Modeling approaches

We distinguish three modeling approaches:

- **Deterministic Models**
  A deterministic model [16, 17] is based upon differential equations which describe the kinematics (or dynamics) of a vehicle. The output of these models is fully determined by the kinematic equations and the initial conditions. Deterministic models assume that the states of a vehicle is perfectly known and the differential equations are perfect representation of the vehicle's motion. However, such models do not consider uncertainties in the vehicle's states, for example due to the sensor noise. This significantly limits the utility of deterministic models.

- **Stochastic Models**
  Stochastic Models do not assume that vehicle states are deterministic, but rather take into consideration uncertainties on the vehicle state and its evolution. Gaussian Noise Simulation and Monte Carlo Simulation are two common stochastic modeling approaches used for path prediction [7]. Both these approaches use a kinematics model which represents the expected motion of the vehicle.

However, similar to Deterministic Models, Stochastic Models also need to be hand designed based on the expected motion of the vehicle and sensor characteristics. However, it may be difficult to describe vehicle motion in complex situations with motion equations. This limits the complexity of Stochastic models to a designer's understanding of vehicle motion. Furthermore, incorporating new features such as the effect of road curvature on the vehicle velocity may require expensive redesigning of the motion model.

- **Data Driven Model**
  Unlike Deterministic and Stochastic models, Data Driven Models [18, 12] are not based on kinematic equations. Rather, they learn to map a set of inputs to expected outputs, based on training data. Depending on their complexity, they may be powerful enough to generalize from limited training data and learn patterns which are difficult to model with kinematic equations. Furthermore, they are easier to retrain with new data each time an additional feature dimension needs to be added, to model more complex driving scenarios.

  However, these have two limitations; one the data-driven models are not based on kinematic relations which may produce unexpected outputs. Secondly training these models requires a considerable amount of data, which may not be easily available.

For this work, we select two models for vehicle trajectory prediction, a baseline model based on a Bayesian modeling approach and a Data Driven Neural Network model.

3-3 Bayesian Modeling

Bayesian modeling is a widely used approach to track objects, specifically Kalman filtering. A Kalman filter is an stochastic estimator based on a First Order Markov Chain that infers parameters of interest from uncertain observations. It assumes all measurement noise is obtained from a Gaussian distribution.

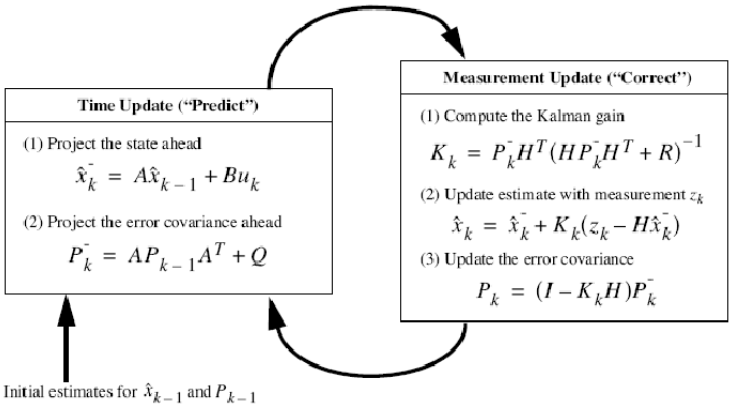


Figure 3-1: Kalman filter equations.

Figure 3-1 gives the equations for a Kalman Filter. Where x is the state vector, u is the input vector, z is the observation vector, A is a the dynamic (motion) model, H is the observation

matrix, B is the input matrix, P is the covariance matrix, Q and R are the Covariances of the process noise and the measurement noise respectively, X^- and P^- are the uncorrected predicted state and covariance, X^+ and P^+ are the corrected predicted state and covariance. Subscript $k-1$ and k represents time steps.

A Kalman Filter (KF) predicts the states of an object using a motion model and then updates its prediction based on real measurements. This gives the best estimate of the object's state at the current time step which is also used as a prior for the next time step. When no measurements are available, the prediction by the motion model can be directly used as the best estimate. This can be used as a prediction model. Thus, initially the model will track the object as long as measurement of the vehicle's states are available, followed by predicting the vehicle's state based on the motion model with the last corrected estimate.

Assuming a point object, constant velocity and constant acceleration are two possible motion types exhibited. Each of these motion types can be modeled by a separate motion model of a Kalman Filter. Equation 3-1 and 3-2 describe the motion equation for constant velocity and constant acceleration Kalman filter respectively.

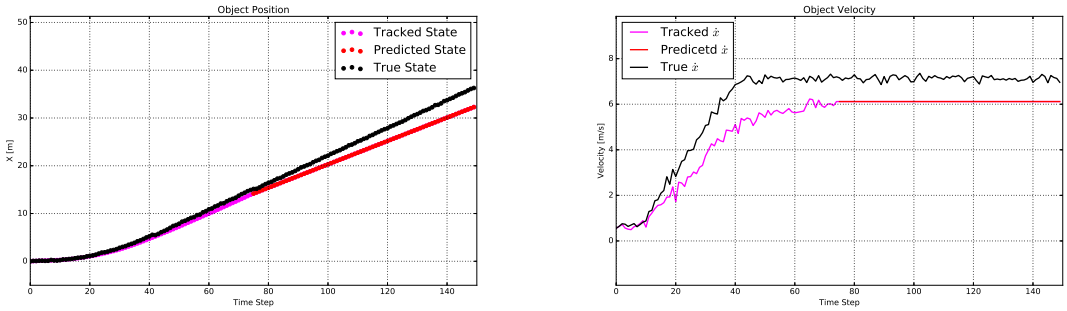
$$\begin{bmatrix} X_k \\ Y_k \\ \dot{X}_k \\ \dot{Y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X_{k-1} \\ Y_{k-1} \\ \dot{X}_{k-1} \\ \dot{Y}_{k-1} \end{bmatrix} \tag{3-1}$$

$$\begin{bmatrix} X_k \\ Y_k \\ \dot{X}_k \\ \dot{Y}_k \\ \ddot{X}_k \\ \ddot{Y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 & 1/2 * dt2 & 0 \\ 0 & 1 & 0 & dt & 0 & 1/2 * dt^2 \\ 0 & 0 & 1 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & dt \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X_{k-1} \\ Y_{k-1} \\ \dot{X}_{k-1} \\ \dot{Y}_{k-1} \\ \ddot{X}_{k-1} \\ \ddot{Y}_{k-1} \end{bmatrix} \tag{3-2}$$

A maneuvering vehicle may change between the two motion types. For such a maneuvering vehicle, a Constant Velocity model based Kalman Filter (CV-KF) can track a vehicle moving with constant velocity well. However, when tracking an accelerating object, CV-KF would need to continuously update its velocity based on the measurements. Although a CV-KF would be able to catch up with changing the value of velocity, it would require several time steps to do so. As a result, the KF would lag. Furthermore, for prediction, when no (velocity) measurement are available, the KF would predict a vehicle's position to increase linearly even if the object was last seen accelerating.

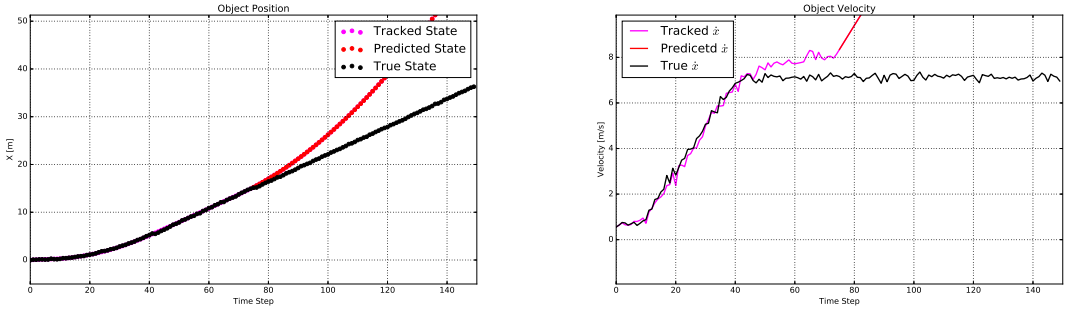
On the other hand, a Constant Acceleration Kalman Filter (CA-KF) would have no lag in tracking an accelerating object. However, it would give a very noisy tracking output when the vehicle moves with (near) constant velocity. This is because small variations in the vehicle velocity, possibly due to noise, would be perceived as acceleration. That is, the CA-KF would have difficulty differentiating between an accelerating vehicle and velocity measurement noise. Furthermore, when predicting, if the model perceives a small acceleration, this would accumulate over time to give a very high velocity and thus a poor trajectory prediction.

Figure 3-2 and 3-3 show tracking and prediction of a maneuvering vehicle using a CV-KF and a CA-KF. The vehicle is initially tracked for the first 3 seconds followed by prediction. For simplicity, 1D motion corresponding to longitudinal motion of a vehicle, is used. The vehicle initially exhibits constant velocity, followed by deceleration and then again constant velocity. Both the maneuvering happened within the duration of the tracking. Gaussian noise is added to the measurements, with variance obtained from LIDAR based object detection.



(a) (b)

Figure 3-2: Tracking and predicting 1D (longitudinal) motion of a maneuvering vehicle using Constant Velocity Kalman Filter. The object is tracked for the first 75 time steps, followed by predicting the trajectory for the next 75 time steps. Each time step corresponds to $1/25$th of a second. a) Vehicle Position b) Vehicle Velocity



(a) (b)

Figure 3-3: Tracking and predicting 1D (longitudinal) motion of a maneuvering vehicle using Constant Acceleration Kalman Filter. The object is tracked for the first 75 time steps, followed by predicting the trajectory for the next 75 time steps. Each time step corresponds to $1/25$th of a second.

Neither KF model is solely suitable for tracking and predicting the trajectory of a maneuvering vehicle, which exhibits a combination of the two motion types. Thus a model is required that switches between the two motion types depending on the current observable motion. Given the fact that the system can be described with one of two well-defined models, an Interactive Multiple Model filter is suitable for this task.

3-3-1 Interactive Multiple Model Filter

An Interactive Multiple Model (IMM) filter consists of a bank of (sub)filters. In our case, Kalman Filters, each designed to perform well for tracking a certain type of motion. Each KF is executed in parallel to track a maneuvering vehicle. The filter output at each time step is obtained using a weighted sum of the output from each KF, where the weights are based on the probability of the models fitting the current motion.

Similar to KF, an IMM is typically used for tracking an object [1, 19]. This work extends IMM for predicting object states.

Working of IMM

An IMM filter can be separated into four steps, Interaction between Kalman Filters, execution of individual Kalman Filters, obtaining model probabilities, and mixing of output from individual Kalman Filters. Figure 3-4, gives a schematic representation of the functioning of the IMM.

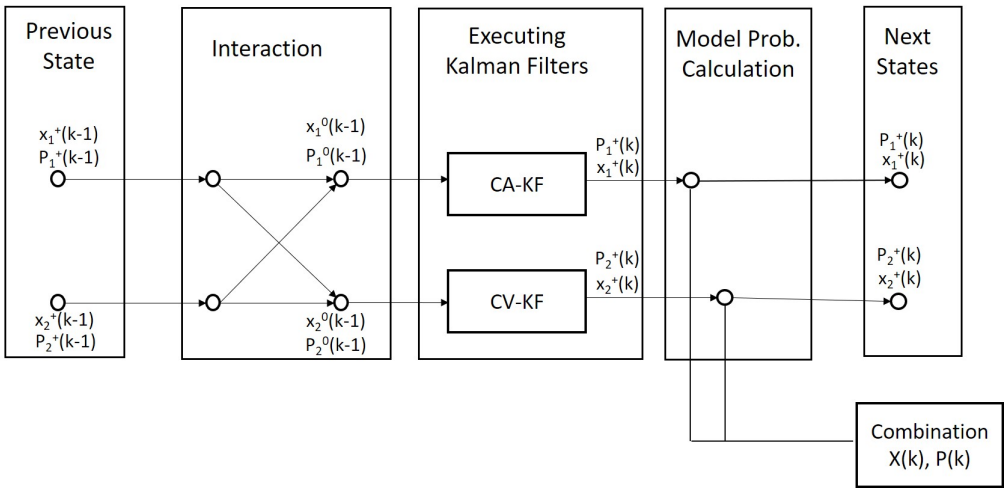


Figure 3-4: Interactive Multiple Model Filter [1]

The first step calculates the mixed states and the covariance by combining corrected output of each Kalman Filter from the previous time step. The second steps involves executing individual Kalman Filters with the mixed state and covariance as the last corrected states and covariance. The third and fourth step calculates the probability of each model representing the current observed vehicle motion, and based on these model probabilities the corrected predicted state and covariance is calculated.

The following describes in detail the four steps[19] : -

- The first step combines the corrected state estimates and the covariances of the individual Kalman Filters from the previous time step. The combination is based on model probability u_i and model transition probability p_{ij}. Model probability u_i is the probability of model i to be best suited for the current object motion and transition probability

p_{ij} is the probability that the object would transition from a model i to model j motion type. Equations 3-3 to 3-5 represent this interaction step between individual KFs.

$$U_{ij}(k-1) = \frac{p_{ij} * u_i(k-1)}{\Sigma_{l=1}^{r} p_{lj} * u_l(k-1)} \tag{3-3}$$

$$X_j^0(k-1) = \Sigma_{i=1}^{r} X_i^+(k-1) * U_{ij}(k-1) \tag{3-4}$$

$$P_j^0(k-1) = \Sigma_{i=1}^{r} U_{ij}(k-1)\Big(P_i^+(k-1) + [X_i^+(k-1) - X_j^0(k-1)][X_i^+(k-1) - X_j^0(k-1)]^T\Big) \tag{3-5}$$

Where i, j, l represent different Kalman Filters, r is the total number of Kalman Filters, subscript k and $k-1$ represent the time stamp, $u_i(k-1)$ represent the model probabilities at previous time step $(k-1)$, U_{ij} is the conditional probability of an object to exhibit motion mode i transiting from motion model j, $X_i^+(k-1)$, and $P_i^+(k-1)$ are the updated state and covariance from the previous time step, while $X_i^0(k-1)$ and $P_i^0(k-1)$ are the mixed state and mixed covariance.

- The second step involves executing the individual KFs based on their motion model represented by equations 3-6 and 3-7.

$$\begin{aligned} X_j^-(k) &= A_j * X_j^0(k-1) \\ P_j^-(k) &= A_j * P_j^0(k-1) * F_j^T + Q_j \end{aligned} \tag{3-6}$$

$$\begin{aligned} K_j &= P_j^- * H_j^T (H_j * P_j^-(k) * H_j^T + R_j)^{-1} \\ X_j^+(k) &= X_j^-(k) + K_j * (Z(k) - H_j * X_j^-(k)) \\ P_j^+(k) &= (I - K_j * H_j) * P_j^-(k) \end{aligned} \tag{3-7}$$

The subscripts j represent the jth Kalman filter. For further details see figure 3-1.

- The third step updates the model probabilities, given by equations 3-8 and 3-9. The model probabilities are based on how likely motion model j is suited for the current object motion. The residue $Z(k) - H_* X_j^-(k)$ provides this measure. A lower residue signifies a better match between object motion and the motion model (KF).

$$\begin{aligned} S_j(k) &= H_j * P_j^-(k) * H_j^T + R_j \\ \hat{y}_j(k) &= Z(k) - H_j * X_j^-(k) \\ d_j^2(k) &= \hat{y}_j(k)^T * S_j(k)^{-1} * \hat{y}_j(k) \\ \Lambda_j(k) &= \frac{exp[-d_j^2(k)/2]}{\sqrt{|2 * \pi * S_j(k)|}} \end{aligned} \tag{3-8}$$

$$u_j(k) = \frac{U_j(k-1) * \Lambda_j(k)}{\Sigma_{i=1}^{r} U_j(k-1) * \Lambda_i(k)} \tag{3-9}$$

Subscript i and j represents the jth Kalman filter, Λ_j is the probability density function of the normally distributed measurement estimate $H_j * X_j^-(k)$ with $Z(k)$ as mean and covariance as $S_j(k)$. $u_j(k-1)$ and $u_j(k)$ are the model probabilities at time step $k-1$ and k

- The final step combines the corrected state $X_j^+(k)$ and covariance $P_j^+(k)$ from individual Kalman Filters are combined using equation 3-10 and 3-11.

$$X(k) = \Sigma_{j=1}^{r} X_j^+(k) * u_j(k) \tag{3-10}$$

$$P(k) = \Sigma_{j=1}^{r} u_j(k)\big[P_j^+(k) + [X_j^+(k) - X(k)][X_j^+(k) - X(k)]^T\big] \tag{3-11}$$

$X(k)$ and $P(k)$ are the combination of states and covariance of r Kalman filter, and output of the IMM.

3-3-2 Using IMM for Path prediction

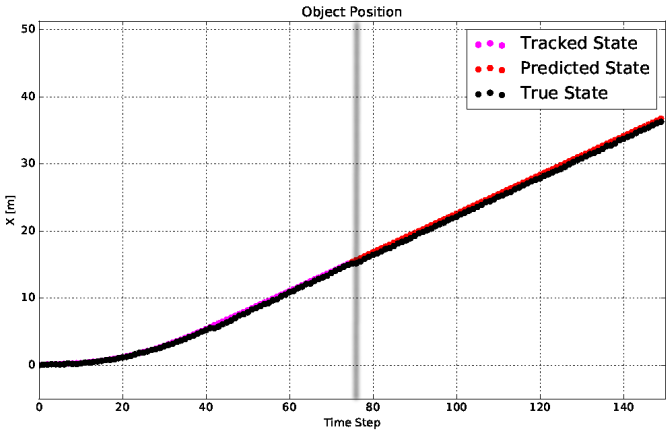
During tracking, the individual Kalman Filters of the IMM are provided with noisy sensor measurements from which they output a corrected state estimate. Based on the residues of the individual KF models, the model probabilities are calculated, which lead to obtaining the mixed states and covariances. When no measurement is available, that is during prediction, the Kalman Filters are simply updated without a measurement, as done with individual Kalman filters. Furthermore, since now it is not possible to obtain a residue to update the model probabilities, the model probabilities remain unchanged from the last calculated value. The prediction is made based on these models probabilities.

Figure 3-5 shows the performance of the IMM filter in tracking and prediction a maneuvering vehicle. The IMM uses the same model of CA-KF and CV-KF, as was used to predict a maneuvering vehicle trajectory in figure 3-2 and 3-3. As compared to a CV-KF, the IMM shows much-reduced lag during tracking. At the same time, the IMM is able to predict the vehicle to move with constant velocity where the CA-KF failed.

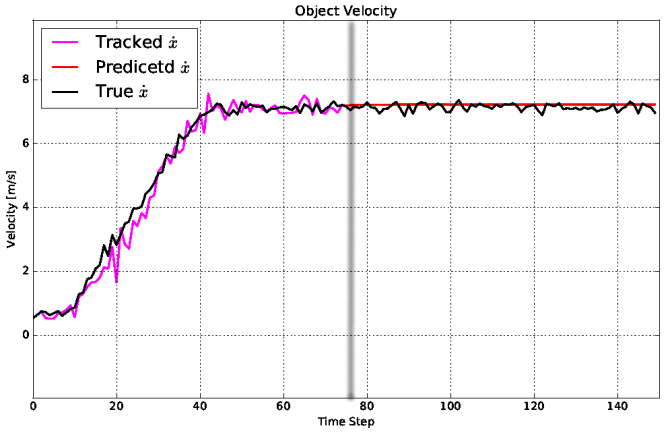
Figure 3-5(c) gives the model probabilities over time, the CA model probability increases when the maneuvering vehicle is observed to decelerate and then reduces back at the end of the maneuver. This switch allows the IMM to track the vehicle well during the maneuver. When no measurements are available to track the vehicle, the last model probabilities are used to predict the vehicle's trajectory based on a two CA and CV motion models.

3-3-3 Conclusion

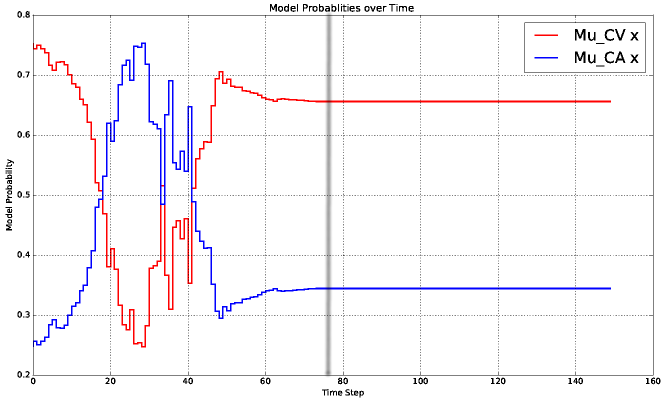
An IMM prediction model can reasonably track and predict the trajectory of a maneuvering vehicle that switches between different motion types. The model is able to extract the vehicle motion behavior during tracking the object. This extracted motion information is then used to predict the vehicles future trajectory.



(a)



(b)



(c)

Figure 3-5: Tracking and predicting 1D (longitudinal) motion of a maneuvering vehicle using the Interactive Multiple Model Filter. The object is tracked for the first 75 time steps, followed by predicting the trajectory for next 75 time steps. Each time step corresponds to $1/25$th of a second.

As the IMM can be used to predict vehicle motion in CCS, this would allow incorporating the road geometry constraint. However, consider a scenario where the vehicle is approaching a curved road section, the velocity of the vehicle would be a function of the road curvature. The higher curvature, the higher reduction in speed, directly affecting the vehicle's motion. The vehicle would decelerate to a minimum velocity proportional to the maximum road curvature and then accelerate towards the end of the curve. To make this association, the prediction model should be able to use the road curvature as an input feature and alter the vehicle's speed proportionally. However, the described IMM model can only function with motion data i.e. the kinematic state of the vehicle and is not designed to directly work with data of any other kind like road curvature. Although, the IMM can be extended to incorporate this input by adding to its motion model; this would require a complex low level mathematical modeling of the vehicle dynamics and the road geometry.

This brings forth another limitation of IMM; to adapt an IMM for more complex modeling, the IMM needs to be carefully redesigned each time. For example considering a vehicle turning behavior (represented by a constant turning motion model), or considering a new feature like the influence of road curvature on the trajectory of a vehicle. Redesigning is often non trivial, making the model inflexible and limiting its application.

3-4 Recurrent Neural Network

Recurrent Neural Networks are 'connectionist' [3] models designed for sequential machine learning tasks i.e. problems whose input and/or output data is sequentially related. For example in Video classification [20], Action Recognition [21] or Natural Language Processing tasks like Machine Translation[22]. Recurrent Neural Networks (RNN) are able to selectively pass information over an arbitrarily long context window, which allows them to establish temporal dependencies between the input and output data.

In the domain of trajectory prediction, RNNs have been successfully used for object tracking [23] with performance comparable to Kalman Filters. Furthermore, RNNs have also shown to perform well in predicting trajectories of pedestrians [24] and predict the future location of vehicles on highways [18] using past trajectories. This makes RNNs a suitable choice to investigate their performance for vehicle trajectory prediction.

3-4-1 Working of a Recurrent neural Network

Figure 3-6 gives the architecture of a standard single layer RNN. The key to the RNN functioning is the hidden state h_t, which is passed forward at each time step. This hidden state is used along with the current input to compute the output. Following set of equations govern the computation of a RNN:

$$\begin{aligned} h_t &= f(W_{hx} * x_t + W_{hh} * h_{t-1}) \\ y_t &= W_{yh} * h_t \end{aligned} \tag{3-12}$$

where x_t is the input, h_{t-1} is the hidden state from previous time step, h_t is the hidden state from the current time step, y_t is the output, f is an activation function, (W_{hx}, W_{hh}, W_{yh})

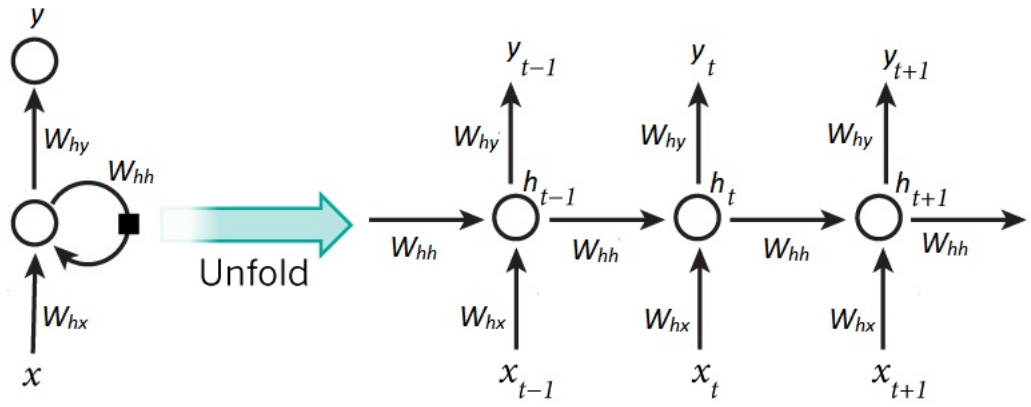


Figure 3-6: Recurrent neural Network [2]

are trainable weights. The weights are trained using a backpropogtion algorithm [25]. Back propagation works in a supervised setting to obtain the derivative of a loss function with respect to each parameter in the network, followed by adjusting the weights by gradient descent to reduce loss.

The hidden states of a network can be thought of as memory of the network, which is used to capture information from the previous time steps. At each time step, the hidden state is updated to include the newly available information. The RNN can capture long range time dependencies, theoretically for any number of time steps [3]. Using this memory, the network can infer sequential relations between inputs over several time steps.

A RNN essentially performs the same task at each time step, that is, produce a desired output based on the current input and the hidden state. This allows the RNN to use the same set of parameters across all time steps, as shown in figure 3-6. This offers multiple advantages, one, the number of parameters to be trained are significantly reduced; furthermore, a RNN can work with data of any length starting at any time step, because it performs the same operation at each time step.

RNN fulfills all criteria for an ideal model. A RNN is non-linear by design as it uses non-linear activations, it operate on sequential data, optimizes its internal weights through supervised learning using backpropogation and can operate on data spanning over various time steps. Furthermore, a RNN model is not explicitly designed to compute motion information and can accept non motion information as well, for example word embeddings in a machine translation task [3].

3-4-2 Sequence to Sequence Model

Figure 3-7 describes four widely used RNN architectures, categorized by length of input and output. For predicting a vehicle trajectory, a RNN would accept past vehicle states along with contextual features over several time steps as input, and output a sequence of predicted vehicle states. That is, the input and output are both sequences of data of different lengths. For a trajectory prediction task, both the input and output for the trajectory prediction tasks are a sequence of temporally related data. As a result a many to many architecture (model d in figure 3-7), also called as sequence to sequence model [22], is best suited.

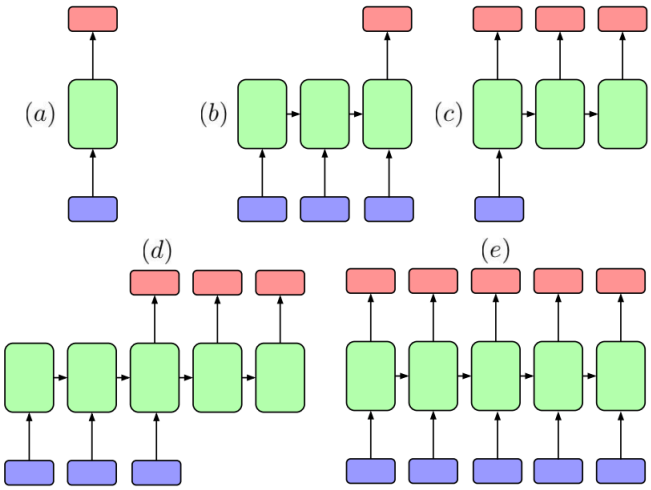


Figure 3-7: Recurrent Neural Network architectures [3].The figure shows different RNN architectures. The blue boxes represents the input, red boxes represents the output and green box are the hidden layers. a) Conventional Neural Network Architecture, b)Many to One architecture, used in text and video classification c) One to Many architecture, used in Image Captioning d) and e) Many to Many architecture, used in machine translation and generative text modeling respectively.

A sequence to sequence (seq2seq) model is specifically designed to map an input sequence data to an output sequence of data. The model consists of two RNNs in series. An encoder RNN reads the input sequence data to obtain a fixed dimensional vector representation in form of the hidden state; the second RNN, the decoder, uses this vector representation to extract the output sequence. The two RNNs, have the same architecture in terms of number of layers and neurons, but do not share the same set of parameters.

Long Short Term Memory

A seq2seq RNN is designed to work with long term dependencies over the input and output sequences. For example, translating a paragraph of words from one language to another. Standard RNN cells, given by the equation 3-14, suffer from the vanishing gradient problem [26] making it difficult to train the neural network on data with long term temporal dependencies. To deal with this, a sequence to sequence model uses Long Short Term Memory (LSTM) [27] units instead of standard RNN units. LSTMs are better able to capture long term dependencies and do not suffer from the vanishing gradient problem.

LSTM cells consists of an additional 'cell' state c_t along with the hidden state h_t, which is transformed using the following set of equations:

$$
\begin{aligned}
i &= \sigma(x_t U^i + h_{t-1} W^i) \\
f &= \sigma(x_t U^f + h_{t-1} W^f) \\
o &= \sigma(x_t U^o + h_{t-1} W^o) \\
g &= tanh(x_t U^g + h_{t-1} W^g)
\end{aligned}
\tag{3-13}
$$

$$
\begin{aligned}
c_t &= f \odot c_{t-1} + i \odot g \\
h_t &= o \odot tanh(c_t)
\end{aligned}
\tag{3-14}
$$

where c_{t-i} and h_{t-1} are cell state and hidden state at previous time step, x_t is the input, $(U^i, U^f, U^o, U^g, W^i, W^f, W^o, W^g)$ is the set of trainable parameters, $\odot$ and σ represents pointwise multiplication and sigmoid nonlinearity respectively.

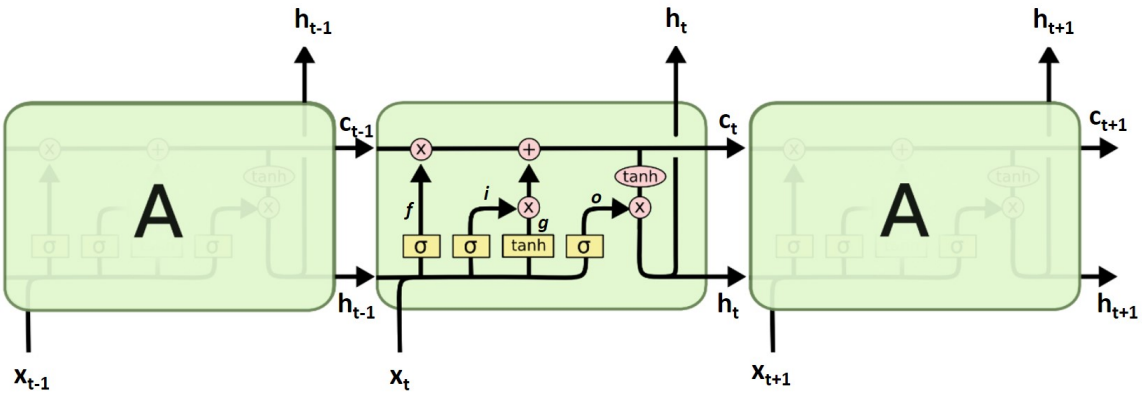


Figure 3-8: Long Short Term Memory cell chain

The cell state c_t is additively altered, based on the values of hidden state h_{t-1} and input x_t. i and f can be described as input and forgets gates, whose value are based on a nonlinear function. The forget gate operates on the cell state from the previous time step c_{t-1} to 'forget' a fraction of value from the previous cell state. The input gate adds to this value to give the new cell state c_t. The hidden state is obtained as a function of the output gate o, and the current cell state. The hidden state h_t is passed to the cell in the next time step as well as output to the higher layer of the network.

In a traditional RNN cell, the hidden state is transformed through a nonlinear operation at each time step. However, in a LSTM, the gated mechanism allows control over how much of the cell state is 'forgotten' and replaced at each time step. As a result, the network can learn to 'remember' certain dependencies over a long period of time by storing it in the cell state, and then forget them when no longer required.

Furthermore, only a fraction of the cell state leaks out at the output h_t of the LSTM cell. This allows the LSTM to keep certain long term dependencies, which are not relevant in the current time step, without affecting the current output.

3-4-3 Sequence to Sequence model for path prediction task

For the trajectory prediction task, the seq2seq model would read the noisy past vehicle states $x_1, x_2 ... x_{t-1}$ along with other non motion features $k_1, k_2 ... k_{t-1}$, and a output sequence of vehicle states $y_t, y_{t+1} ... y_t$, see figure 3-9. The goal of the encoder network would be to determine information about the vehicles motion from the noisy vehicle states and associate them with the non-kinematic features. The information about the vehicle's motion pattern would be encoded in the hidden state of the encoder network, which is then passed to the decoder. The decoder can use this information to predict the vehicle's future trajectory.

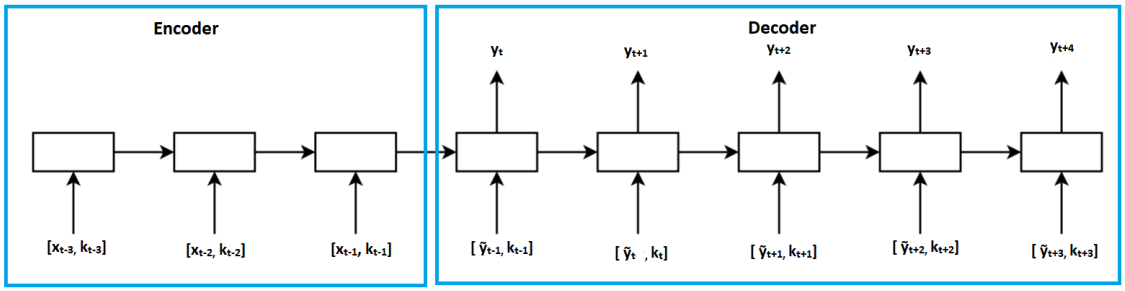


Figure 3-9: Sequence to Sequence Recurrent neural Network

In addition to the hidden state of the encoder, the decoder also requires the input $\tilde{y}_{t-1}$, which represents the vehicle state at each time step, and corresponding the non motion feature k_t. $\tilde{y}_{t-1}$ can be obtained in two ways, either as the output y_{t-1} of the model itself or as the true vehicle state at the time t. During training, when true (future) vehicle states are available, they can be used as decoder inputs. During deployment, when true (future) vehicle states are not available, the model output are used as decoder inputs.

However, this causes discrepancy in training and inference, which leads to poor performance [28]. To deal with this, [28] provides a scheduled sampling training. By this approach, during training, the model output is selected as $\tilde{y}_{t-1}$ with a probability η. That is, if $\eta = 1$, the network is trained before, and is $\eta = 0$, the network is trained with the same setting as inference.

The seq2eq model used for path prediction differs from the model deployed for machine translation [22] in an important way; the Seq2seq model for machine translation maps each word from one language to one or more words in another, That is there is a direct mapping between an input and output value. However, for path prediction, there is no direct mapping between an input and an output, rather the output sequence is a continuation of the input series. This leads to two design changes in the model:

1. Consider a machine translation task in which both input and output sequence is 10 words long. Since there is a direct mapping between input and output value, for each word the corresponding output would be 10 time steps ahead. As result, the LSTM cells

will have to remember data for each word for 10 time steps. To tackle this, [22] reverses the order of the output, that is the sentence is reversed. As a result, the last input can be first translate and the first input last. Now in case of 10 input and output words, the LSTM cells would only need to remember semantics of one word for 10 time step, for the rest words the 'remembrance ' horizon is reduced. This improves the performance of the network in the machine learning task. However, since in a path prediction task the input and output are a continuous series, the output sequence is not revered.

2. For machine translation, the first input to the decoder is a special token, which corresponds to the end of the input series. This token in the input sequence maps to an end of the sentence in the output sequence, which is also represented by the same token value. However, in trajectory prediction task, there is no specific end to a trajectory like a language sentence, rather the input and output form a continuous trajectory. As a result, no such special token is used, rather, the last vehicle state (and non-kinematic features) is used as first input to the decoder.

When deploying a seq2seq RNN on an autonomous vehicle for trajectory prediction, each time an additional state x_t of a surrounding vehicle is observed, the encoder can process this data along with hidden state h_{t-1} to obtain a new hidden state vector. This vector can then be used by the decoder network to predict future vehicle states. Furthermore, this data driven model can be further optimized on the go, i.e. the neural network can be further trained on new data observed by the autonomous vehicle. This can be done one by predicting a vehicle trajectory, and then obtaining a loss by observing the vehicle's real trajectory over next few seconds.

3-5 Conclusion

To predict the trajectories of vehicles operating in its vicinity, an autonomous vehicle requires a model that can extract information about the behavior of these vehicles. Based on the extracted behavior, the future trajectory of vehicles can be predicted. We selected two modeling approaches, a stochastic approach - Interactive Multiple Model Filter and a data driven modeling approach - Recurrent Neural Networks.

Interactive Multiple Model Filter is a tracker based on Bayesian modeling and is extended for prediction task. However, the IMM has shortcomings, specially the inability to work with non-kinematic features. This limits its prediction ability, as the prediction would solely be based on motion features. The RNN was selected to improve upon the limitations of IMM. A Sequence to Sequence architecture with LSTM was found to be the best suited RNN design for path prediction. Such a model can learn to obtain information from input trajectory and output a predicted trajectory based on both past vehicle motion and contextual features. The neural network is trained in a supervised manner using backpropogation algorithms.

The ability of an RNN to associate the vehicle's motion with contextual features is key. This allows the model to anticipate the changes in the vehicle motion during prediction. For example, using the road curvature, RNN can anticipate slowing down of a vehicle on turns. Furthermore, a RNN offers flexibility, as the network can be retrained each time a new contextual features is added, to provide a better predictive ability.

Chapter 4

Experiments

The contributions of this work consists of two parts; First, the use of Road Structure as an Infrastructure Indicators in trajectory prediction. Second, a modeling approach for trajectory prediction using these Infrastructure Indicators along with Motion Indicators. These contributions are evaluated in this chapter.

4-1 Dataset

The dataset used for the experiments consists of vehicle trajectories with road infrastructure information. The data is collected from the test vehicle - WURbie, in the region of Wageningen, The Netherlands, under natural driving conditions. The test vehicle is equipped with 6 IBEO LUX LIDAR sensors, each with a 110 degree (horizontal) FOV and 4 vertical planes. IBEO sensors are installed on the vehicle, three facing forward and three backward. Data from these LIDAR sensors is used to detect, classify and track vehicles moving around the test vehicle.

Figure 4-1 shows the road sections for which the vehicle trajectories are recorded. The road sections are selected such that the only external factors affecting the driving is the road structure itself. Features like Pedestrian crossings, complex road design like roundabouts, traffic signals are avoided, because this work does not model the effects of these features on driving behavior.

The test vehicle has a maximum operational speed of 25 km/h, which affects other vehicles operating in the vicinity especially on roads with higher speed limits. To prevent any variation in behavior of vehicles due to presence of the test vehicle, the test vehicle was parked on the road side such that the LIDAR sensors had a good view on the road length.

The LIDAR sensors provide vehicle position (X, Y), velocity $(\dot{X}, \dot{Y})$ and heading angle θ in the WEpods (ego-vehicle) frame. The vehicle states are convert to the Global coordinate - Universal Transverse Mercator (UTM) coordinate system, using the ego-vehicle state (position, velocity and heading) obtained from GPS based localization. The LIDAR and Localization module are unsynchronized, as a result the (time-wise) closest ego-vehicle state is

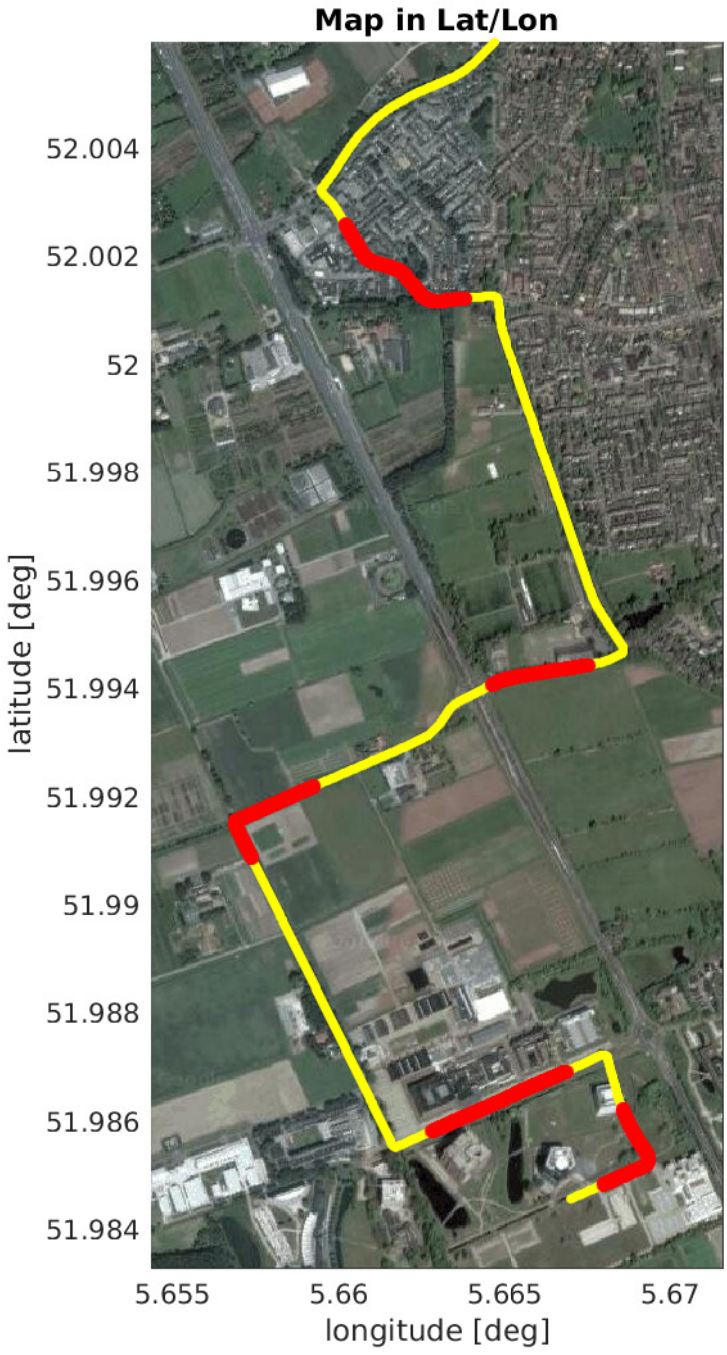


Figure 4-1: Figure shows road sections (in red) on which vehicle trajectories are recorded

used for each LIDAR measurement. Of the five states, only the position and velocity is used for prediction, as heading is found to be inaccurate and very noisy.

4-2 Experiment 1

The Curvilinear Coordinate System (CCS) provides the road geometry constraint, as described in chapter 2. This allows a prediction model to consider the roadway structure, which is lacking in the Global Cartesian Coordinate System. This experiment examines the improvement obtained in trajectory prediction when predicting using vehicle motion description in CCS.

Therefore, we compare trajectory prediction done using the vehicle states described in GCCS and CCS. We predict a trajectory in both coordinate systems and then plot them in GPS coordinates. The prediction model in either case was provided with solely motion information, one described in GCCS and the other in CCS. The model is provided with 2 seconds (50 time steps) of vehicle states as input data and predicts the trajectory for next 8 seconds (200 time steps). The Interactive Multiple Model Filter is used as the prediction model.

For the prediction in CCS, the input trajectory is first converted to CCS from GCCS, prior to the prediction. Post prediction, the obtained vehicle states are transformed back to GCCS, and then to GPS coordinates.

Figures A-1, A-2, A-3 shows the position and velocity plots of three (real) examples of vehicle trajectories predicted in GCCS and CCS. In the first the trajectory, fig. A-1(a), the road makes a sharp 90 degree turn, and in the second figure A-2(a) the roadway makes two consecutive turns of about 30 and 45 degrees. Finally, the third figure A-3(a) gives a trajectory of a vehicle on a straight road.

4-3 Experiment 2

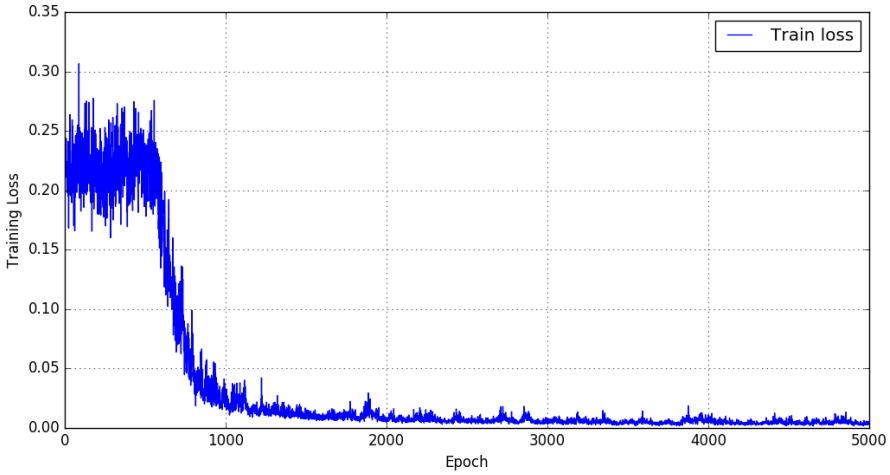
Before, applying the seq2seq model to path prediction, an experiment is designed to establish that the model can satisfactorily perform regression in a non-linear space. Sine-wave prediction is chosen as the regression task. The space is single dimensional, and describes the value of the function $a * sin(2\pi ft + \phi)$. Where, a is amplitude,f is frequency, ϕ is phase and t is independent variable.

The training dataset consists of randomly generated sine waves with random amplitude, frequency and phase. The first 25 samples of the wave are provided as input to the model, which then predicts the next 25 samples. To make this prediction, the three variables a, f, ϕ need to be estimated internally by the model.

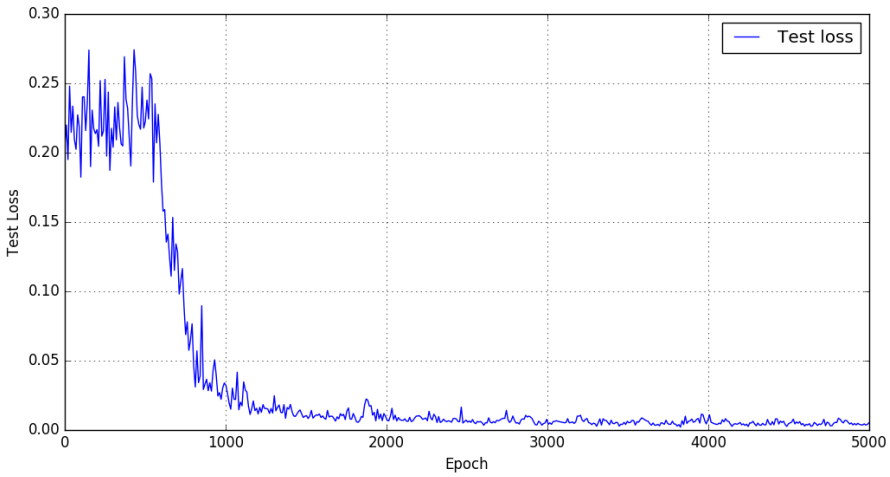
The seq2seq model consists of one hidden layer of 40 LSTM cells. The neural network is trained using Adams optimization [29], with Mean Squared Error loss. Figure 4-2, shows the loss over training epochs for the training and test set [1].

Figure A-4 shows some predicted sin wave from test set samples.

[1]The test loss is calculated after every 10 epochs



(a)



(b)

Figure 4-2: Train and Test loss over training epochs. Minimum training loss: 0.001938, Minimum test loss: 0.003538

4-4 Experiment 3

The third experiment compares the performance of an RNN against an IMM model, including the RNN's ability to exploit Road Curvature as an additional feature. For this we train two RNNs, one with only motion features (described in CCS) as input, and another with both motion features as well as road curvature. The performance of these two and IMM is then compared.

To make the performance comparison, the models are provided with input data for 25 time steps (1 second), and the error is reported for the predicted vehicle position at 25, 50, 100 and 150 time steps. All the trajectories are described in the Curvilinear Coordinate System. Any error in velocity would be reflected in position, as a result the velocity predictions are not scrutinized separately.

The trajectory data is segregated into a training and test set, with 4:1 ratio. Following this, trajectories with a length of 175 time steps (25 + 150) are extracted for each recorded trajectory. For trajectories of longer duration, multiple trajectories are extracted such that each new trajectory starts after 75 time steps of the beginning of the previous trajectory. This is done separately for both training and test set. This provides 496 trajectories for training and 143 for testing.

For all trajectories, the X positions are normalized by subtracting the initial x position. For Y position and the velocities, their absolute value is of significance. The Y position typically does not vary more than the road width. Furthermore the value of velocity on X is directly affected by the curvature of the road. Thus these three values are not normalized.

4-4-1 Training RNN

Two seq2seq RNNs are trained with the only difference of road curvature as input feature. We call the RNN model without road curvature 'Motion RNN', and the other model is called 'Curvature RNN'. Both RNNs have an output dimension of 4 $(X, Y, \dot{X}, \dot{Y})$. For Curvature RNN, since the input and output have different dimensions (the network does not output the road curvature), the decoder network is provided with the road curvature as an auxiliary input at each time step

The seq2seq network is designed to have one or more hidden LSTM layers. The output layer consists of (four) standard RNN cell with Rectified Linear unit (ReLu) non-linearity [2], that are unbounded in first quadrant. This is because, the desired outputs are not bounded, for example a vehicle may potentially travel a distance in range of [0-inf] during period prediction, depending on the vehicle speed and the duration of prediction.

To address the fact that only a limited amount of data could be collected, we compared the two training regimes; One where the network is directly trained with data collected from test vehicle. Another setting where an RNN is first trained on artificial data and then trained on real data (transfer learning). The artificial data consists of vehicle trajectories emulating lane following behavior on straight as well as curved roads. Using a simple dynamic modeling of a vehicle on a curved road section, the reduction in longitudinal velocity of the vehicle was

[2] ReLu activation represented by $f(x) = max(x, 0)$

simulated. Gaussian noise was added to all states, with mean and variance obtained from LIDAR measurements. More details on the artificial data set can be found in appendix B.

Training the network with the transfer learning regime gave an 8% improvement in performance. We used this network for further results. We used Adams [29] optimization algorithm to train the network. The loss is calculated using as Mean Squared Error over the four output states $(X, Y, \dot{X}, \dot{Y})$. The network is trained with a constant learning rate of 10^{-3}.

4-4-2 Results

To evaluate the performance of the models, we segregated the test trajectories into three groups based on road curvature values. The first subset of trajectories are vehicle paths on near straight roads, the second subset consists of curved road sections with small curvature values (< 0.1). Finally the last subset consists of vehicle trajectories on sharp curves, that is higher curvature values (> 0.1).

Figure 4-3 gives the error in meters on X and Y axis for the three models - IMM, Motion RNN, and the Curvature RNN with all three test scenarios. The error is reported after 1, 2, 4 and 6 seconds of prediction. Appendix A gives additional plots to compare the three models.

Table 4-1 and 4-2 gives the mean error for the three models.

	IMM	Motion RNN	Curvature RNN
1 sec	0.95	0.82	**0.87**
2 sec	1.90	1.78	**1.55**
4 sec	4.87	3.98	**3.19**
6 sec	9.47	7.61	**6.20**

Table 4-1: Mean error [m] on the X axis for all test trajectories

	IMM	Motion RNN	Curvature RNN
1 sec	0.70	0.73	**0.80**
2 sec	1.39	0.86	**0.90**
4 sec	2.91	1.24	**1.30**
6 sec	4.48	1.33	**1.38**

Table 4-2: Mean error[m] on the Y axis for all test trajectories

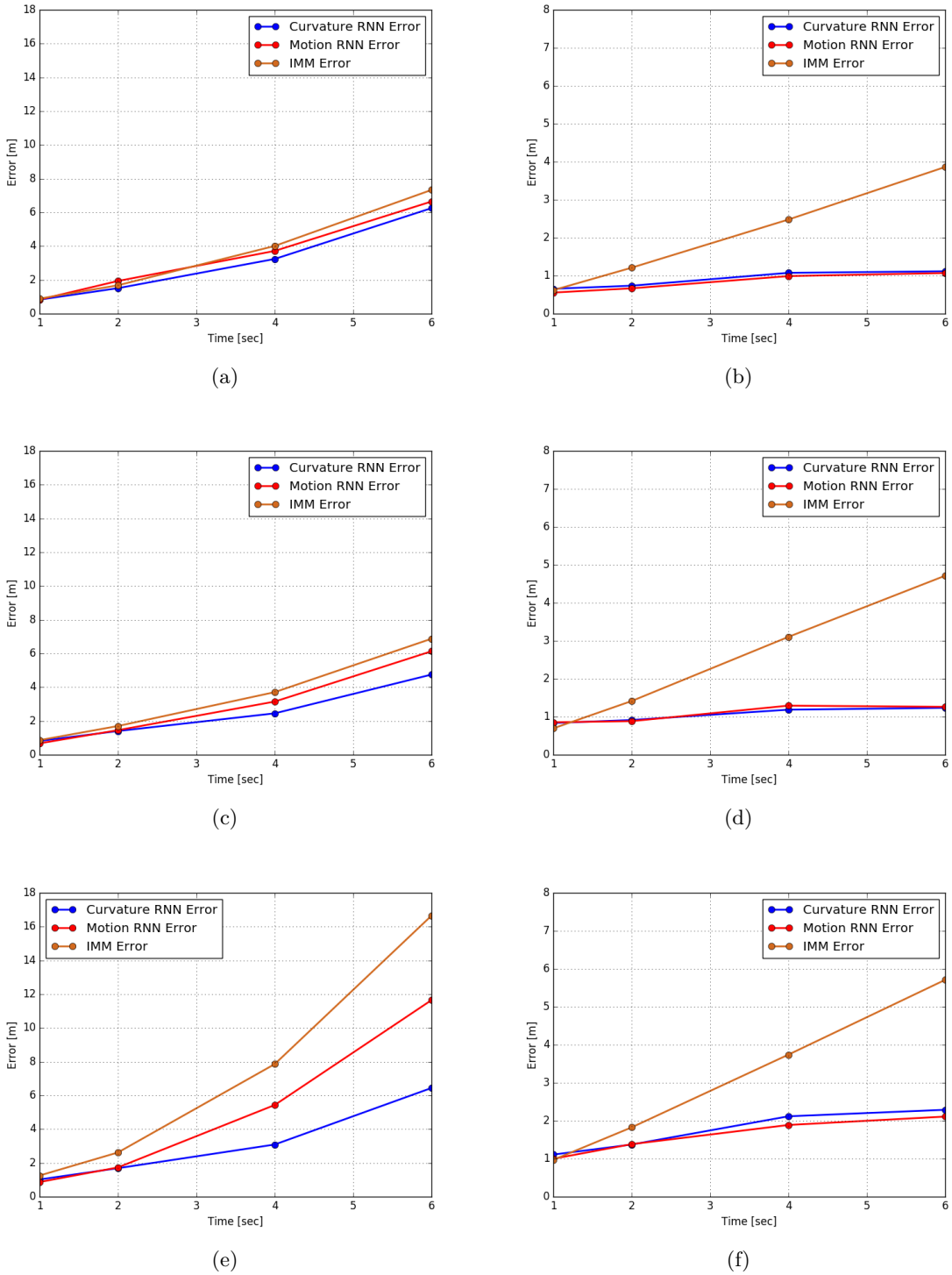


Figure 4-3: Prediction Error (in m) on the X and Y axis for test trajectories over prediction time. Figure a) and b) show error on the X and Y axis for straight road trajectories (curvature $<$ 0.001), figure c) and d) show error on the X and Y axis for roads with 0.001 $<$ curvature $<$ 0.1, figure e) and f) show error on the X and Y axis for high curvature roads (curvature $>$ 0.1)

4-5 Experiment 4

In this experiment we examine the effect of varying the length (number of time steps) of the input data on Sequence to Sequence RNN model's performance. For this, separate RNNs are trained with training data of an input length of 5, 12 , 25 and 50 time steps. The performance of each of these RNNs is reported for test trajectories with 5, 12 , 25 and 50 time steps of input length. The performance is measured by error in X and Y position after 6 seconds of prediction. The RNNs are provided with both past vehicle states as well as road curvature information as input. The network parameters (weights and biases) used are the same as obtained in experiment 3 for Curvature RNN model.

The same training and test set (in 1:4 ratio) are used in each setup, with the difference that the total trajectory length in each setup differs. Tables 4-3 and 4-4 show the error of each trained RNN on data with varying input length.

Input length during Testing

Input length during Training	**5**	**12**	**25**	**50**
5	6.25	24.26	32.64	38.36
12	26.60	6.17	11.70	16.25
25	21.05	7.13	6.16	7.04
50	40.96	23.0	15.57	6.16

Table 4-3: Error on X axis [m]

Input length during Testing

Input length during Training	**5**	**12**	**25**	**50**
5	1.22	1.33	1.47	1.71
12	1.58	1.29	1.34	1.54
25	1.33	1.27	1.35	1.52
50	1.75	1.49	1.60	1.56

Table 4-4: Error on Y axis [m]

Table 4-5: Performance of Sequence to Sequence Recurrent neural Network with varying number of time steps in input data

Chapter 5

Discussion

This work aims at improving the accuracy of vehicle path prediction over a longer duration of time in complex urban settings. Conventional methods use past vehicle motion information to learn the vehicle behavior, and predict only based on motion information. This allows to predict a linear future trajectory. However, vehicles rarely exhibit linear motion.

Non-linearity in vehicle motion is caused by external factors, of which the Road structure is an important cause. The road structure causes non-linearity at two levels, one due to variation in velocity direction as the road itself changes direction, and second, variation in vehicle speed to safely negotiate a curve. This work uses Infrastructure Indicators to linearize both these non-linearities in form the Curvilinear Coordinate System and Road Curvature.

For making the prediction itself, using motion and road structure information, two models are selected - the Interactive Multiple Model Filter and the Sequence to Sequence Recurrent Neural Network. The IMM is used as a base model whereas the RNN is selected as the advanced model, to improve upon the limitations of the IMM.

5-1 Curvalinear Coordinate System

Trajectory prediction using the motion description of a vehicle in Global Cartesian Coordinate Systems offer no information about the road structure. As a result the prediction model is solely based on the vehicle's motion. This doesn't affect the prediction on a straight road, as the road does not significantly change direction and the vehicle state evolves almost linearly in X and Y, as seen figure A-3(a). However, when predicting the trajectory of a vehicle on a curved road section, the vehicle's motion is strongly conditioned by the road structure and is no more linear in X and Y. This results in poor prediction, when using motion in the Global Coordinate System (GCCS) for curved road sections, as seen in figure A-1(a) and A-2(a).

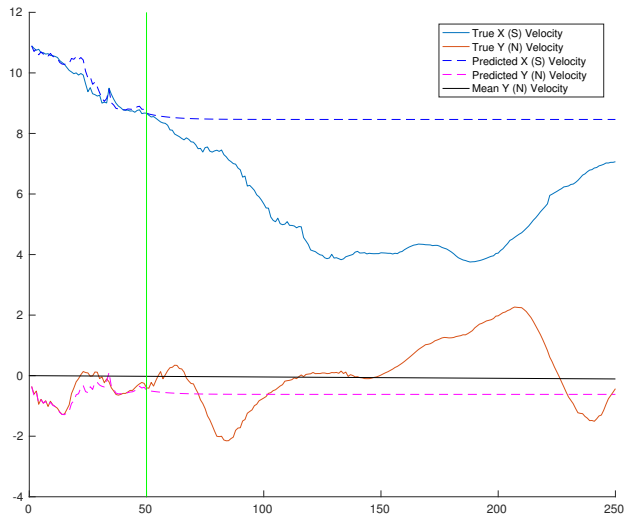
Alternatively, the Curvilinear Coordinate System describes vehicle motion as a function of the road structure. This eliminates the non-linearity in the vehicle trajectory caused by the curved road section. As a result the trajectory prediction becomes less complex for

the prediction model. When transformed back to GCCS, the trajectory prediction is more accurate. Essentially, the transformation of motion from GCCS to CCS linearizes the effect of road geometry on vehicle state. This linearlization effect can be observed in figure 2-7 and 2-5. When converting vehicle states back to GCCS, the non linearity is reintroduced. The effect of predicting in CCS is vizualized in figure A-1(a), A-2(a) and A-3(a). In the case of predicting a vehicle trajectory on a straight road, the two predicted trajectories are overlapping and CCS does not provide an advantage. However, on curved roads sections, prediction in GCCS is unable to keep up with the curved roadway geometry. The model predicts the vehicle to have a linear motion, and the trajectory goes off road, whereas when predicting in CCS, the predicted trajectory is non linear and evolves along the roadway geometry. This can also be visualized from velocity plots A-1(b) and A-2(b). In GCCS, the IMM model predicts the velocity to evolve linearly, whereas the ground truth trajectory is only linear for a very short duration, and then evolves non-linearly due to change in the direction of the road. CCS adds this non-linearity to the predicted motion. Without the road geometry constraint in CCS, the trajectory prediction is accurate for a short duration of time until the vehicle exhibits linear motion (locally). Thus, by adding the road structure through CCS, we assure better prediction over a longer duration of time by allowing prediction of non linear vehicle motion. This already provides a considerable improvement over predicting a vehicle trajectory using only motion information. Interestingly, the velocity plots (A-1(b) and A-2(b)) show that even though the predicted velocity is non-linear in GCCS, its profile is very different from the ground truth. The predicted velocity quickly changes between time step 100 and 150, however, the ground truth velocity changes with much lower acceleration over a longer duration of time. Comparing with motion in CCS, see figure 5-1(a), this difference in the velocity profiles can be associated with reduction in the velocity along the S axis.

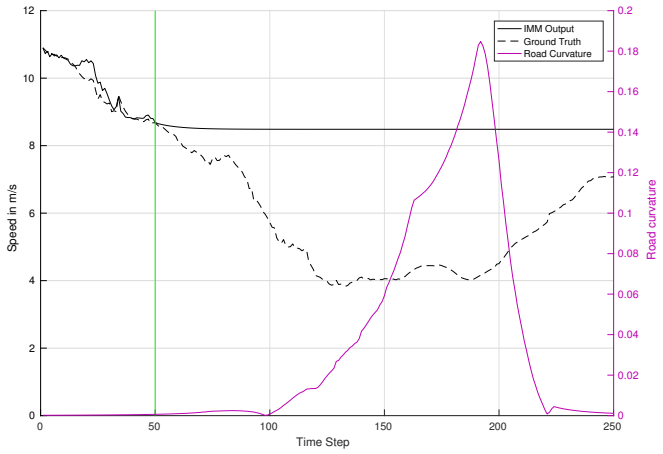
By associating the vehicle's longitudinal velocity with the road structure (see figure A-1(a)), it can be observed that the non-linear motion is caused when the vehicle takes the curve. To better understand this behavior of the vehicle we plot the vehicle velocity and curvature on a graph in figure 5-1(b). The vehicle lowered its speed as it approached the curved section and then raised the speed again towards the end of the curve. However, the CCS itself does not provide any information about the road curvature value. As a result the prediction model has no means to anticipate this non-linearity in motion due to the road curvature. That is, to the prediction models, the space appears to be linear which leads to poor prediction. Higher curvature causes more non-linearity in the prediction space. The prediction can thus only be improved, by further linearizing the space using the feature dimension which adds the non-linearity i.e. road curvature.

5-2 Models for Path prediction and effect of Road Curvature Feature

Recurrent Neural Networks provide the ability to linearize the prediction space using non-kinematic features such as road curvature. This linearization allows improved trajectory prediction. Figure 4-3(a),4-3(c) and 4-3(e), show the performance of three models, IMM,



(a)



(b)

Figure 5-1: a) Predicted and ground truth vehicle velocities on a curved road b) Predicted and ground truth longitudinal vehicle velocities plotted alongside road curvature.

Motion RNN and Curvature RNN on different road curvatures. The effect of using road curvature as additional input to RNN can be best observed on the road section with high curvature (figure 4-3(e)). Curvature RNN gives an average error of 6.44 meters in longitudinal direction after 6 seconds of prediction, which is nearly twice better than Motion RNN (gives an error of 11.65 meters) that does not use the road curvature as a feature. Furthermore, of total 21 test trajectories, 20 were better predicted by Curvature RNN, with up to 15 meter less error.

The mean velocity for the 21 test trajectories is 8.7 m/s, which means on average the vehicles traveled about 52 meter during 6 seconds of prediction. The Curvature RNN gives an error of about 12% after 6 seconds of prediction, vs 22% for Motion RNN. This proves the road curvature was indeed the cause of the non-linearity in the motion, which the Curvature RNN is able to linearize and hence make better predictions.

Figure A-6 and A-7 show the improvement in prediction by Curvature RNN. In both trajectories, the vehicle travels on roads with a sharp curve, which causes reduction in longitudinal velocity of the vehicle. Curvature RNN is able to anticipate this behavior using the curvature information. Furthermore, Curvature RNN is also able to predict a rise in longitudinal velocity as the curvature value reduces.

The difference in error on Y axis between Curvature RNN and Motion RNN is insignificant. From trajectory A-6 and A-7, there are small deviations in the vehicle's lateral positions as the vehicle negotiates the curve. However, the Curavture RNN does not learn this behavior but rather predicts the vehicle's lateral velocity to be near a mean value of zero, similar to Motion RNN.

The improvement obtained by using road curvature as an Infrastructure Indicator tops on the improvement in prediction performance by the RNN models itself over IMM. For all three scenarios of varying curvature, RNN performs better than IMM. The difference in the error on X axis increases with higher curvature values. This is because the RNN learns the driving behavior like a vehicle typically does not drive for significant duration at very low (longitudinal) velocity such as 1m/s (see figure A-8), or if a vehicle's velocity oscillates, it would resettle close to its previous stable value A-10.

Importantly, significant improvement is obtained in (lateral) Y axis error, when using the RNN model, over the IMM. This is observed for all three scenarios. (Motion) RNN gives 3 to 4 times less error than IMM, see figure 4-3. If the IMM observes the velocity on Y axis to accelerate, or even have initial constant positive or negative velocity, the IMM predicts the vehicle to continue this behavior for the duration of the predictions. This can be seen figure A-10 and A-7. However, we know that, even a small constant velocity on the Y axis would very quickly lead the vehicle off road. A vehicle would make a counter maneuver in opposite direction to prevent going off-road. The IMM is not able to capture this behavior, and gives a higher error which increases linearly with time. However, the RNN picked up this behavior from training; that a vehicle typically does not exhibit either a positive or negative velocity on the Y axis for any significant amount of time. From figure A-10 and A-7 it can be seen that the RNN rather predicted the velocity close to zero (mean), which causes a much smaller error than IMM. The error on the Y axis for RNN initially increases, and then becomes constant (see figure 4-3). The RNN learnt to keep the the Y axis position bounded, that is the vehicle will not typically go off road. This is a big improvement over IMM.

Overall within CCS, for curved roads we obtained an error of 12% (6.44m over 6 seconds) by using Curvature RNN, which is 3 times better than 32% (16.65 m) over using IMM, on the X axis. Furthermore, an error of 6.44m is in the vicinity of a typical car length (about 4.5m). However, an error of 16m is considerably larger and can make the difference between planning a safe or unsafe path by an autonomous vehicle. Similarly, the error on Y axis is 3 times less for Curvature RNN than for IMM. This, coupled with the improvement obtained by the linearization effect using CCS, makes trajectory prediction considerably better.

5-3 Deployablity

In real driving scenarios, it maybe required to predict a vehicle's trajectory within a small duration of first observing the vehicle. Also, a vehicle's state over longer duration of time maybe available to models it's behavior and predict a likely future trajectory.

Table 4-3 and 4-4 show the performance of the Curvature RNN model for predicting trajectories with different length of input. Error in prediction at the 150th time step (after 6 seconds of prediction) is nearly the same for each RNN when each RNN is trained as well as tested with the same lengths of input trajectory. However the performance deteriorates as the number of input time steps are increased or decreased.

The model is capable of predicting well with different lengths of input, when the training and test samples have similar input lengths in all four setups. However, when the input is significantly of different length during testing, the encoder network of the RNN is unable to extract features that represent the vehicle behavior. It can thus be inferred that the encoder network is not able generalize over different length of input data. This is understandable, because the network did not see any sample during training other than a fixed input length.

The encoder network of the Sequence to Sequence model repeats its operation each time a new input is observed, using the same set of weights and biases. As a result, it is possible to train the network with training data of different input lengths. This is would allow the encoder to better generalize over different lengths of input and extract features from the input trajectory which best represent the vehicle behavior.

5-4 Limitation of RNN

The RNN provides the ability to exploit non-kinematic features such as road curvature, that gives a significant improvement over the IMM. However, unlike IMM, the RNN is not based on a kinematic model to track and predict the vehicle states. Rather, the RNN learns its internal parameters to best match the output values observed during training. As a result, the position and velocity generated as the output of the neural network are not related by the vehicle kinematics i.e. $x = v * t$. This can be observed in figure A-8. Both Motion RNN and Curvature RNN predict a very similar longitudinal velocity profile, however, the end position of the trajectory predicted by the Curvature RNN is much closer to the ground truth than the end position predicted by the Motion RNN.

To better visualize this, we calculate an alternate velocity prediction by differentiating the position output, as well as an alternate position by integrating the output velocities. We

call these 'Alternate Position' and 'Alternate Velocity'. Figure A-16 plots these along with the RNN output positions and velocities, for the same trajectory as A-8. The Alternate Position prediction is very different from the output position prediction, the same applies to the Alternate Velocity Prediction.

Interestingly, during deployment the decoder network of the Sequence-to-Sequence model uses the velocity outputs from the previous time step directly as input for current time step. This means, at each time step the velocity and position predicted at previous time step is available to the RNN. However, the network predicts the position based on an alternate velocity rather than based on the (decoder) input velocity itself. This alternate velocity can be thought of as a cell state in the LSTM cells that is not directly output. This shows that the network is unable to learn the kinematic relation between position and velocity. Rather maintains a separate cell state for each output independently, based on which the outputs are calculated.

This lack of kinematic modeling is an important shortcoming of data driven models like RNN, compared to stochastic models that have these relations built in by design. To improve this shortcoming of RNN, an additional term in the loss function that enforces the kinematic equation $x = \int u(y)dt$ is required. However, this still does not ensure the network will learn the kinematic relation.

5-5 Future work

A Sequence-to-Sequence model was able to work on a non-linear space (in CCS) using the road curvature information, to already providing a significant improvement over IMM as well as Motion RNN. However, the road curvature is only one of many dimensions which add non-linearity to a vehicle's trajectory. For example, the recorded data used for training is obtained from observation of vehicle's on roads with different speed limits. This directly affects the maximum speed. A vehicle on a 25km/h road would typically not exceed this speed limit. However, on a 40km/h road, a vehicle can be expected to reach higher speeds. Inclusion of this knowledge, i.e. the maximum permissible speed for the road section would thus provide a better predictive ability to the model.

Another cause of non-linearity in vehicle motion is the presence of intersections. Vehicles slow down as they approach intersections, and speed up while driving away from them. This is encountered in a small portion of the trajectories collected. To predict this behavior, a feature that describes the vehicle distance from the intersection can be provided as input to the RNN. Of course, to enable a neural network to learn the association between these features and the vehicle kinematic states, a larger amount of training data is required.

With more data, the RNN can be trained to model more complex maneuvers. For example, stopping of a vehicle for a pedestrian crossing. With a correct set of features like distance to the pedestrian crossing as input to the RNN, such complex maneuvers can be predicted. Furthermore, RNNs predicting trajectories of individual vehicles can be made to interact with each other to model vehicle interactions. For example, a reduction in speed of a fast moving vehicle due to a slower moving preceding vehicle. This flexibility to adapt the model to include more features without explicit low level modeling is a key advantage. Thus allowing to model more complex driving scene and predicting accurate vehicle trajectories.

Chapter 6

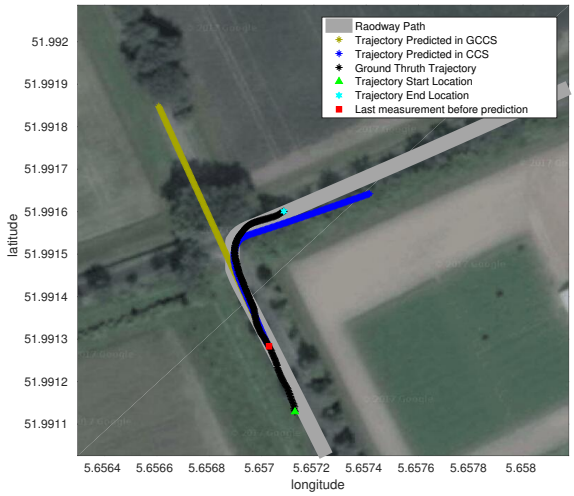
Conclusion

The aim of this work was to improve vehicle path prediction on curved road sections by incorporating road structure as Infrastructure Indicator along with Motion Indicators. Our approach to use a Curvilinear Coordinate System significantly improved the prediction performance by predicting vehicle trajectories along the road geometry. Furthermore, we model road curvature as a feature for predicting velocity variation, as a vehicle negotiates a curved road section. Using this as feature we trained a Sequence-to-Sequence Recurrent Neural Network with trajectory data collected from a test vehicle. The neural network extracts information on the vehicle's behavior from an observed trajectory and predict its future trajectory considering the influence of non-kinematic features such as road curvature. Experiments performed on collected trajectory data show that this prediction model gave three times less error in longitudinal driving direction when using road curvature as a feature. Our prediction model was also able to predict a vehicle to stay within the road width. Thus, this work achieved a significant performance improvement over the existing trajectory prediction approaches for curved road sections.

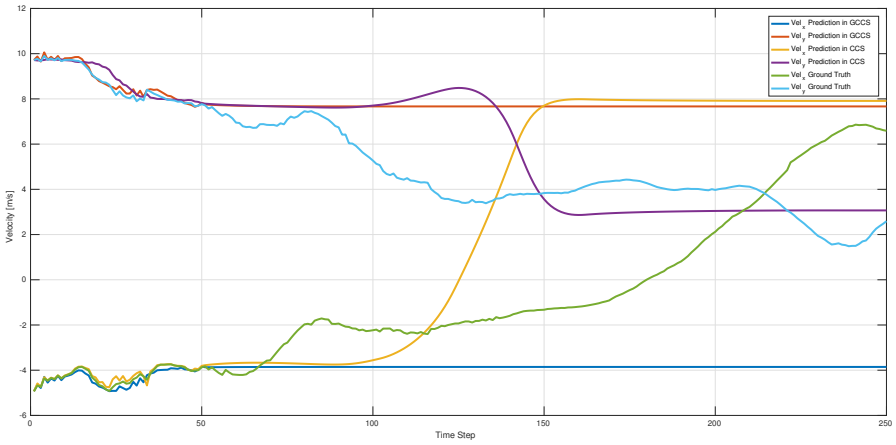
Appendix A

The Back of the Thesis

A-1 Results of Experiment 1



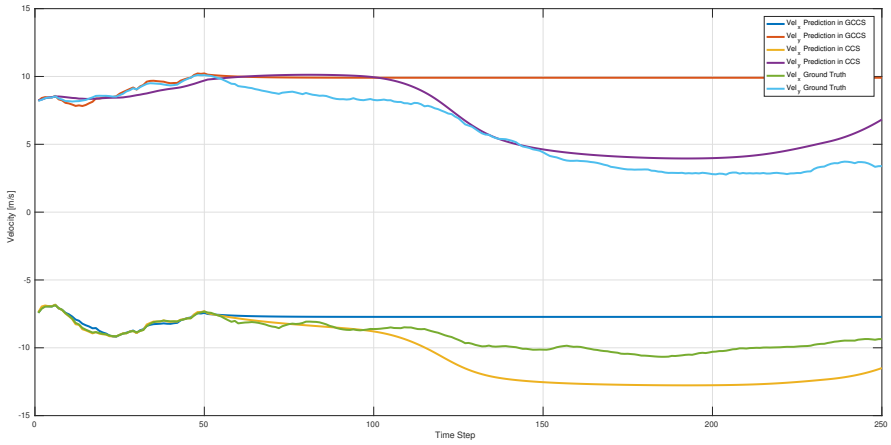
(a)



(b)

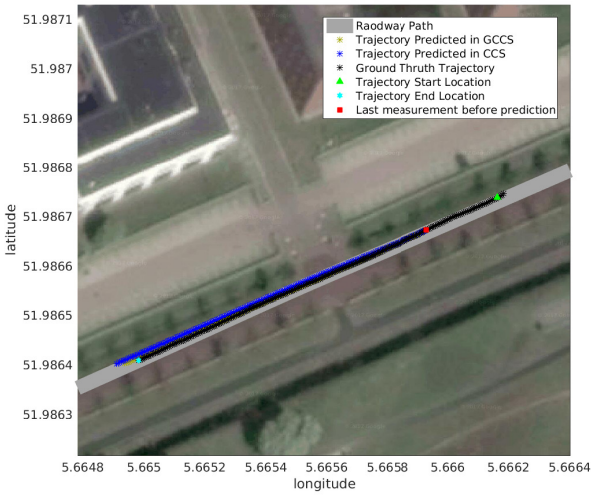
Figure A-1: a) Vehicle trajectories predicted in GCCS and CCS b) vehicle velocities predicted in GCCS and CCS

(a)

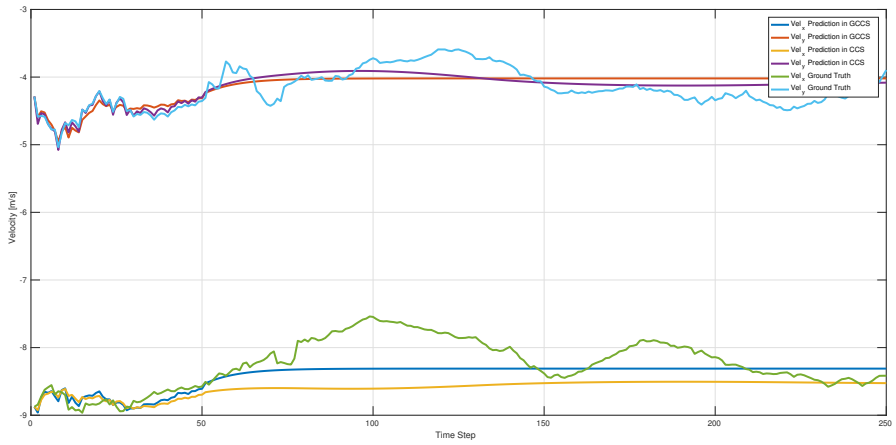


(b)

Figure A-2: a) Vehicle trajectories predicted in GCCS and CCS b) vehicle velocities predicted in GCCS and CCS



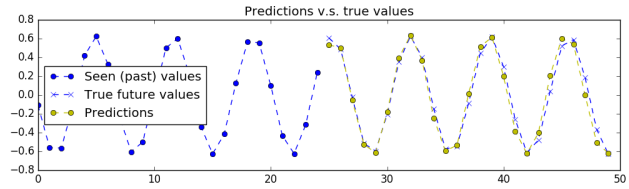
(a)



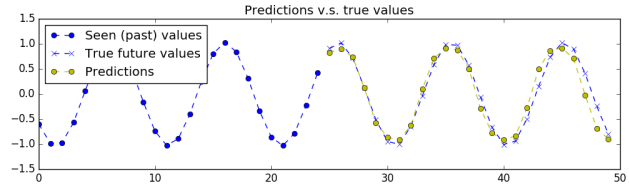
(b)

Figure A-3: a) Vehicle trajectories predicted in GCCS and CCS b) vehicle velocities predicted in GCCS and CCS

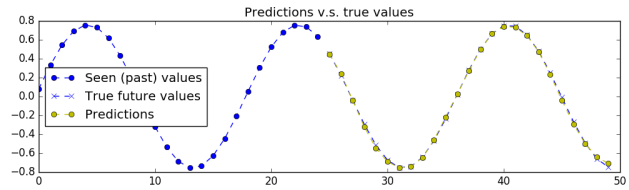
A-2 Results of Experiment 2



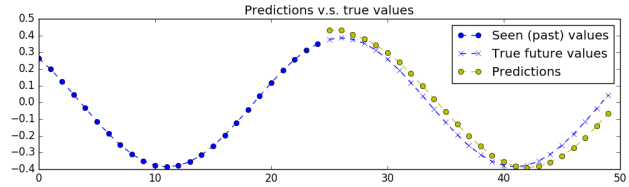
(a)



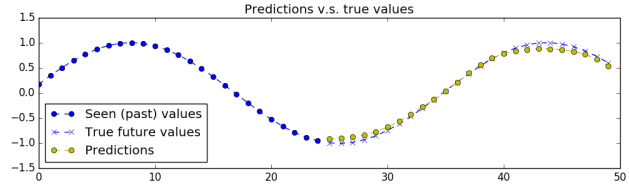
(b)



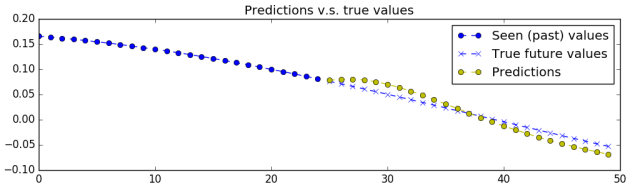
(c)



(d)



(e)



(f)

Figure A-4: Prediction sinwaves from test set

A-3 Results of Experiment 3

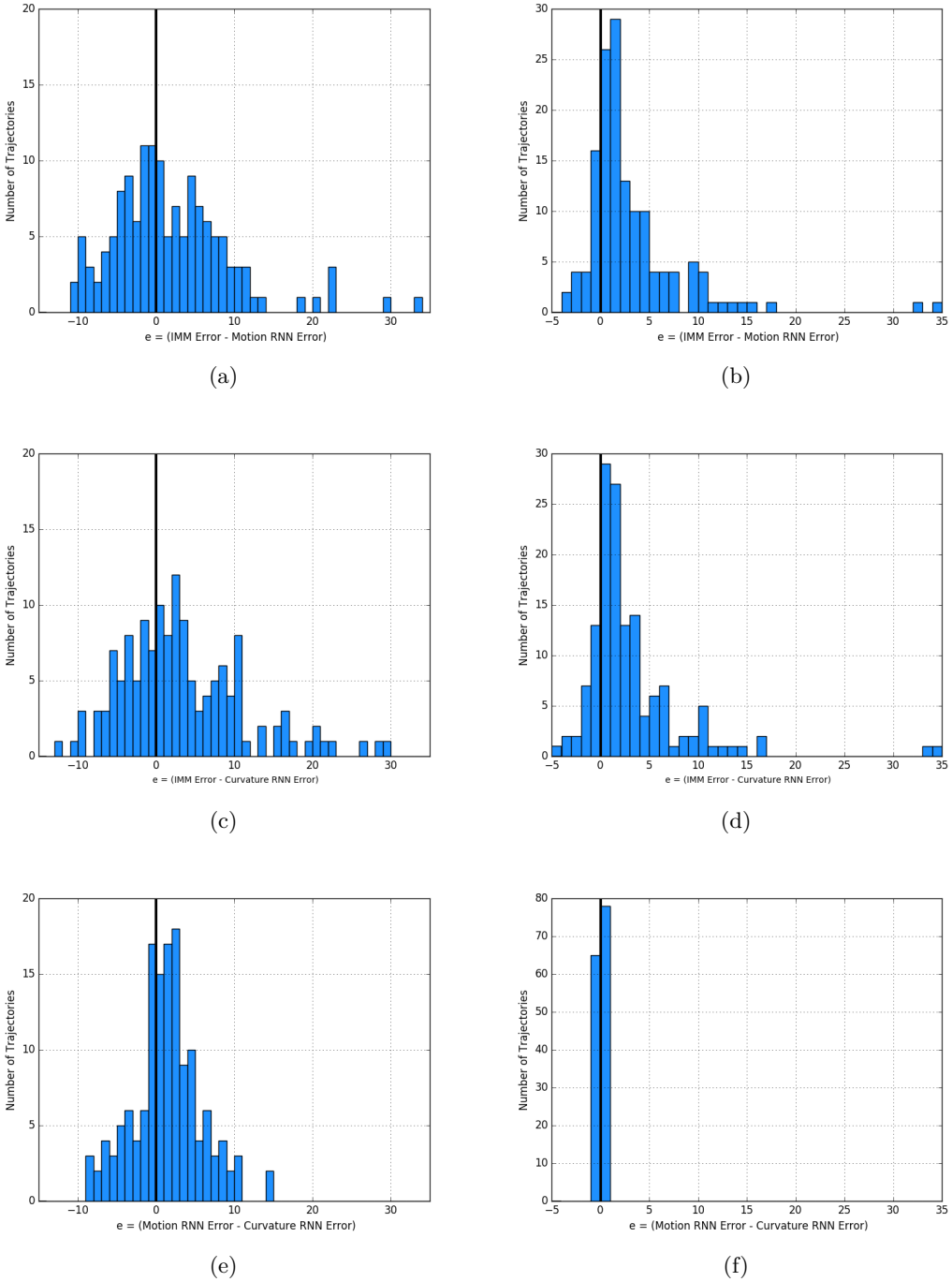
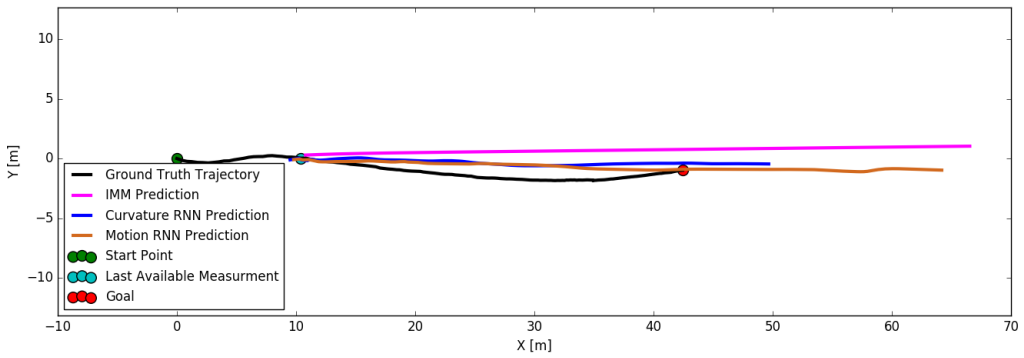
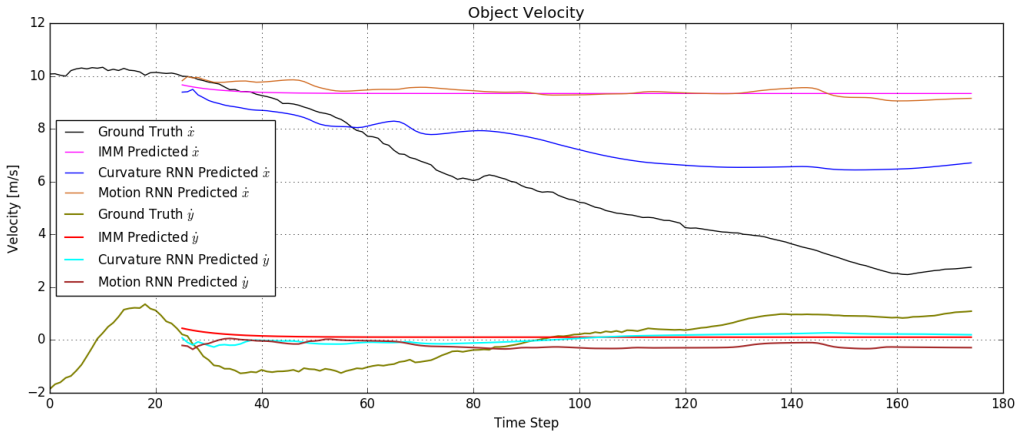


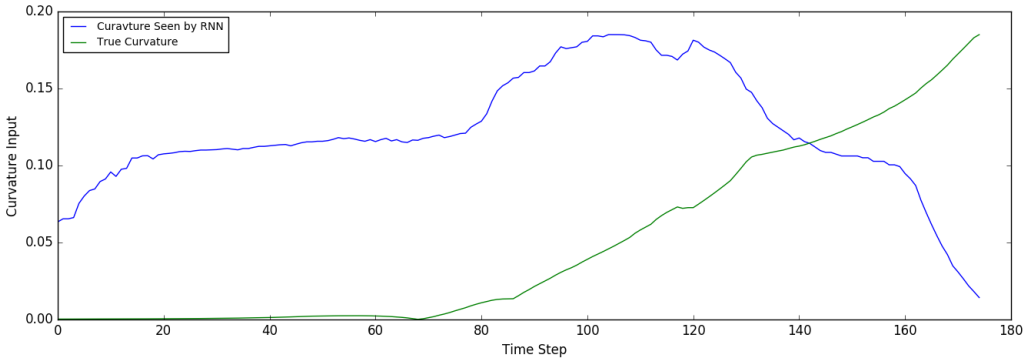
Figure A-5: Comparison of error for all test trajectories predicted using the three models - IMM, Motion RNN, Curvature RNN



(a)

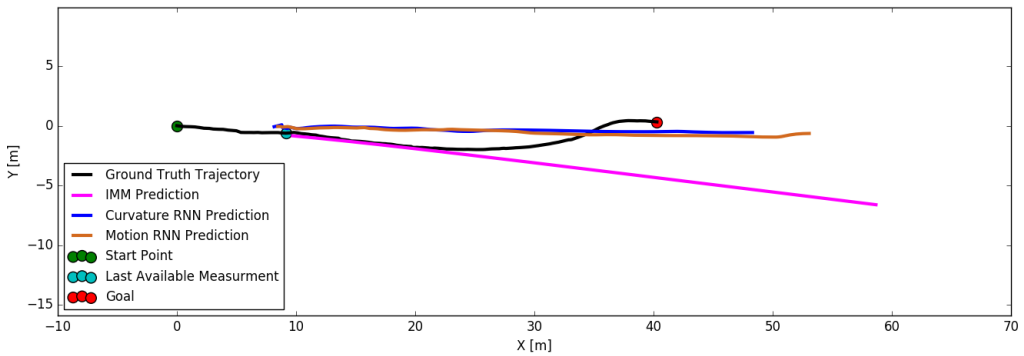


(b)

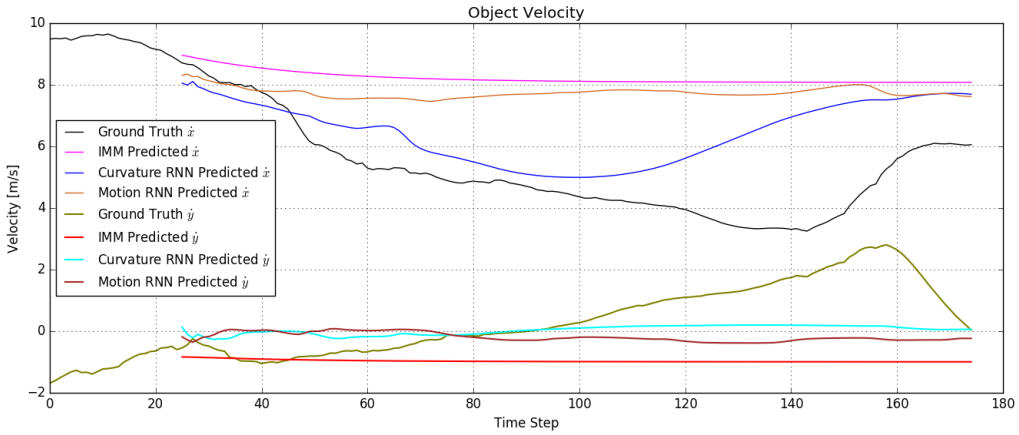


(c)

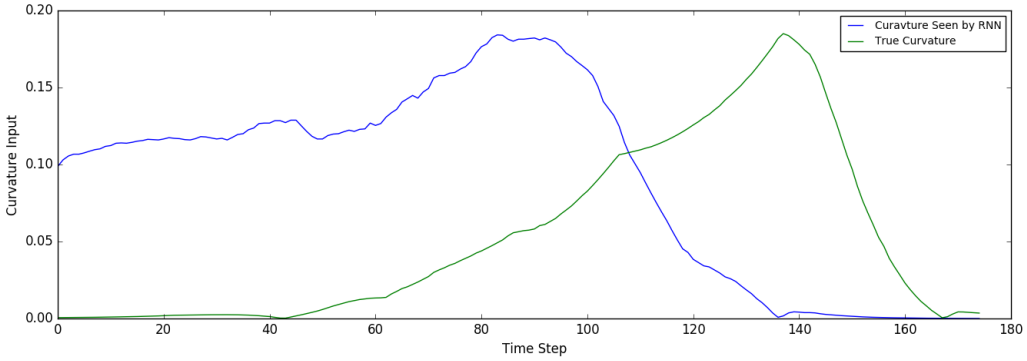
Figure A-6: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature



(a)

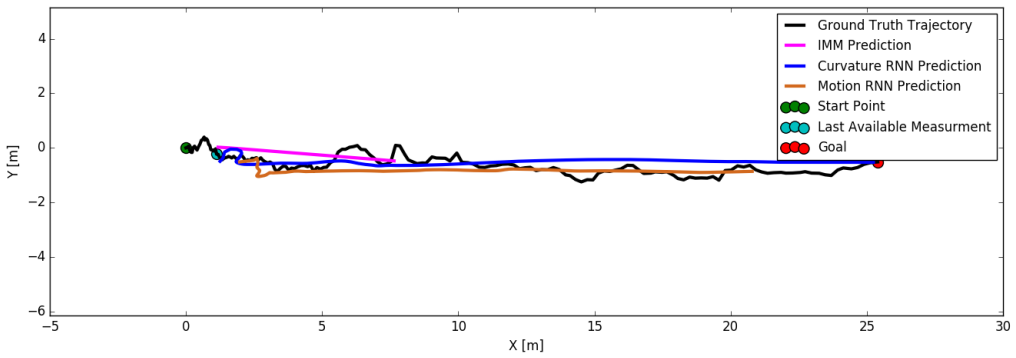


(b)

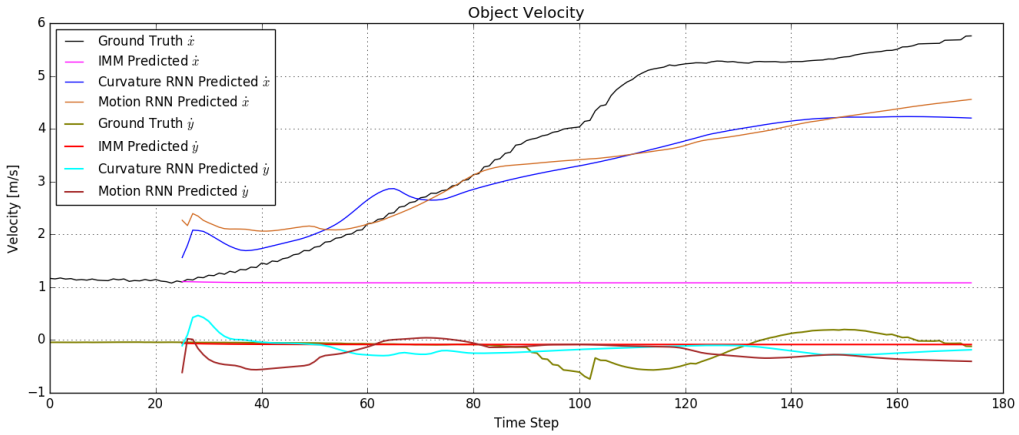


(c)

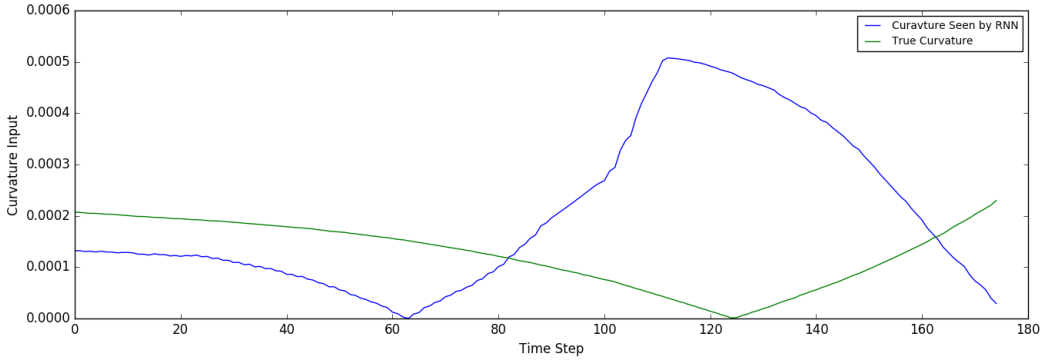
Figure A-7: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature



(a)

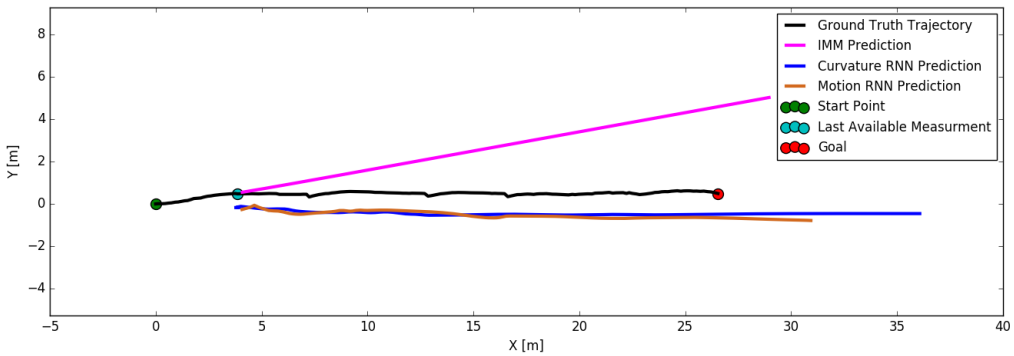


(b)

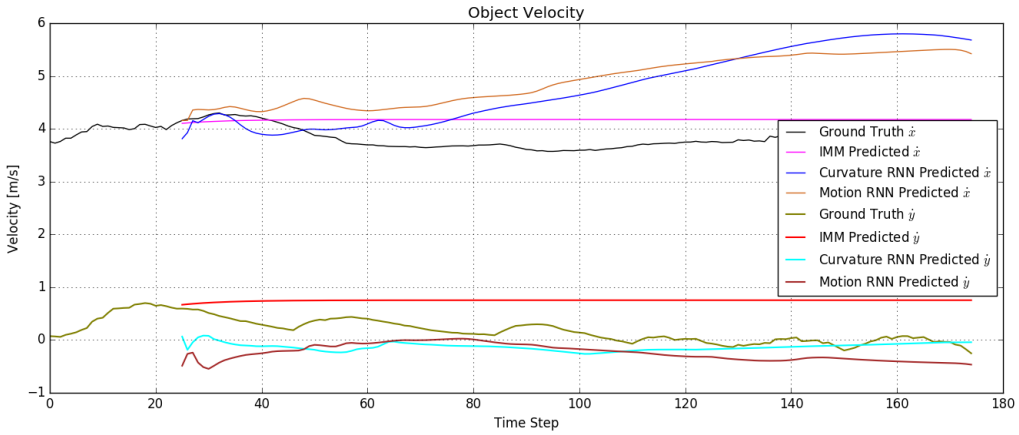


(c)

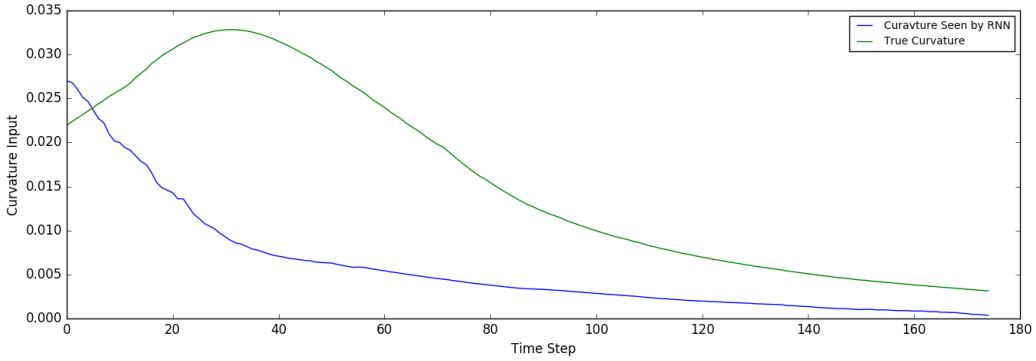
Figure A-8: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature



(a)

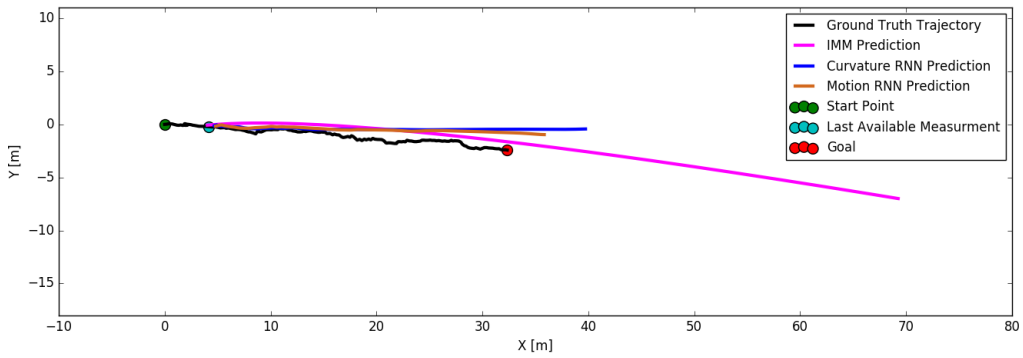


(b)

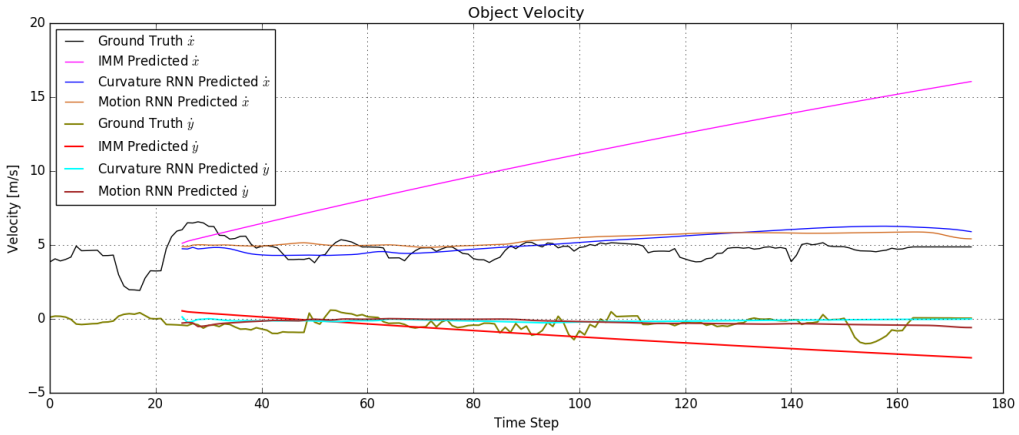


(c)

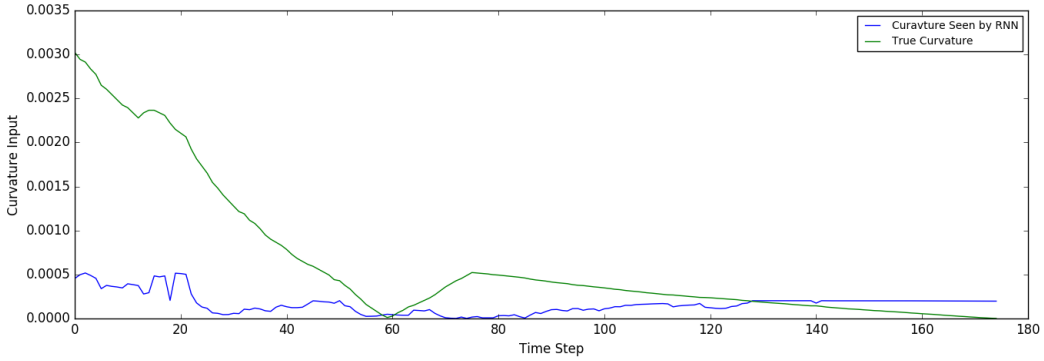
Figure A-9: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature



(a)

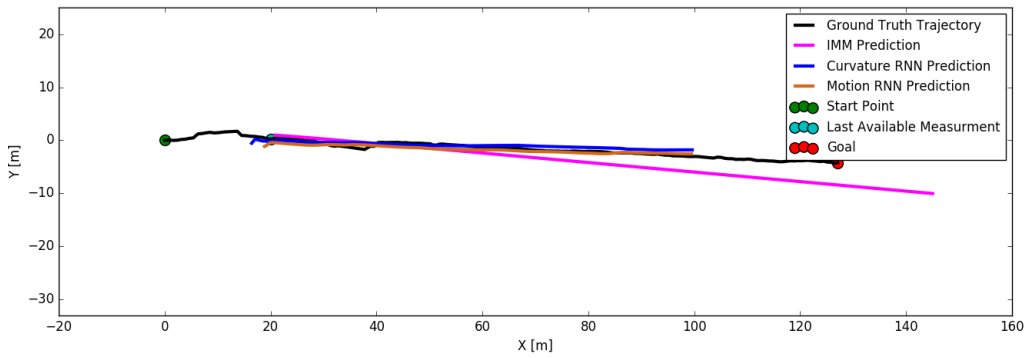


(b)

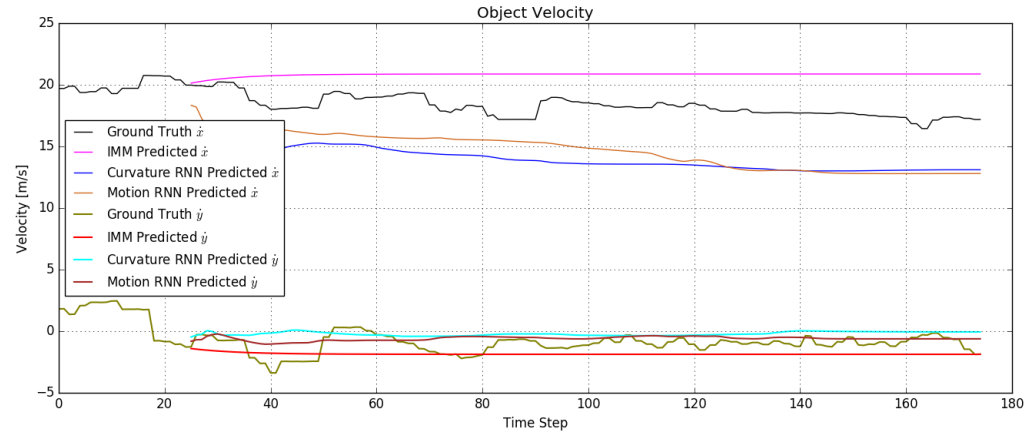


(c)

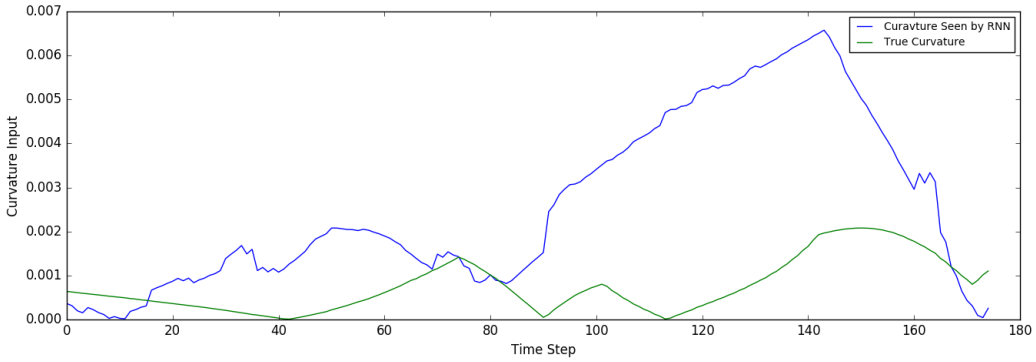
Figure A-10: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature



(a)

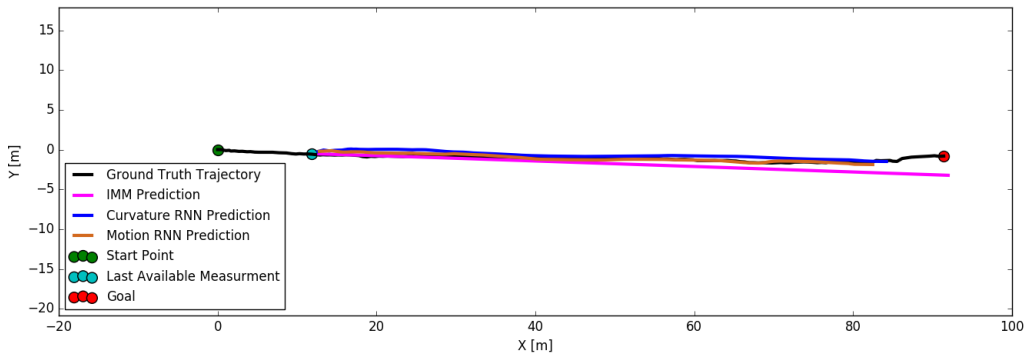


(b)

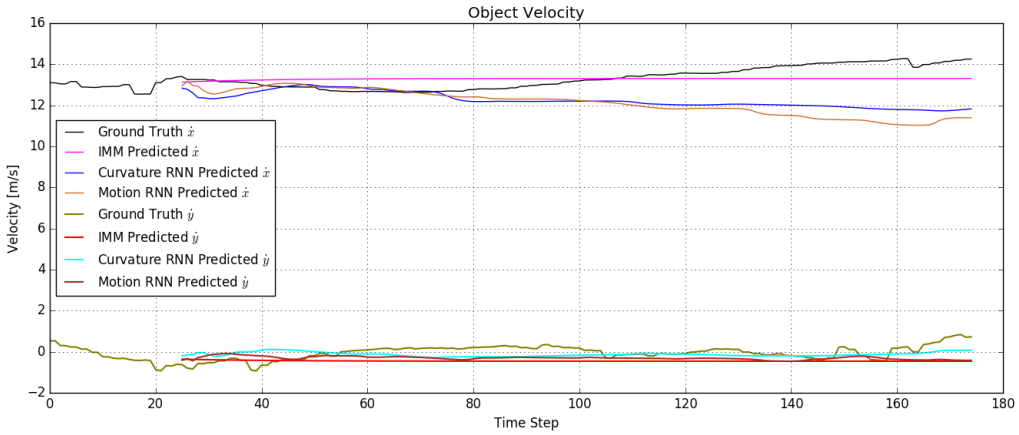


(c)

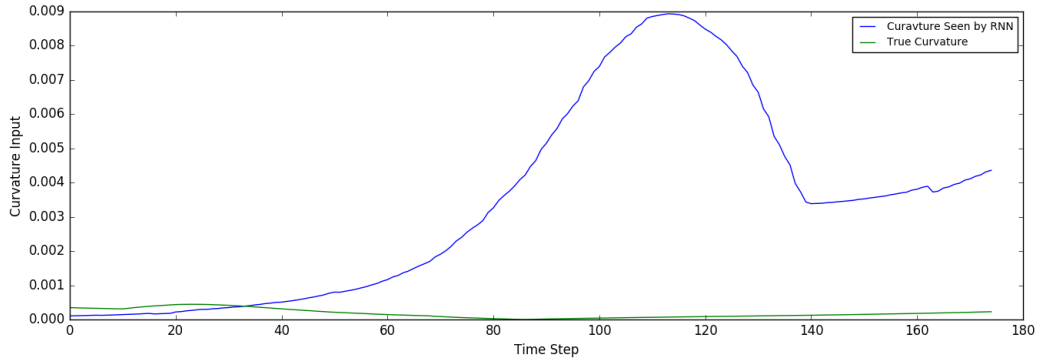
Figure A-11: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature



(a)

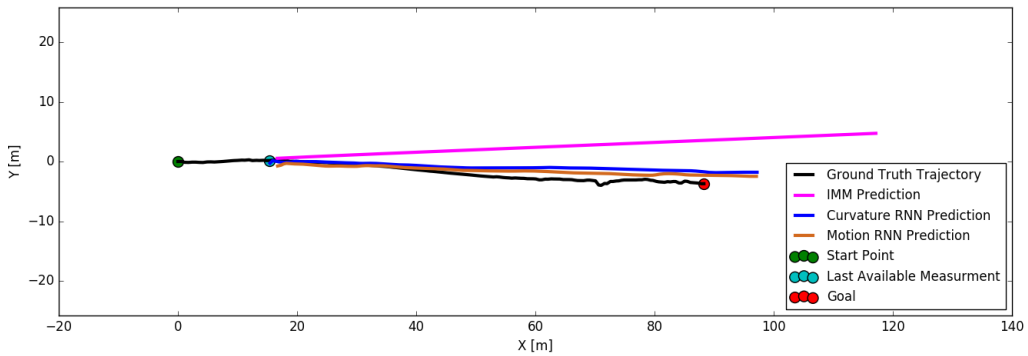


(b)

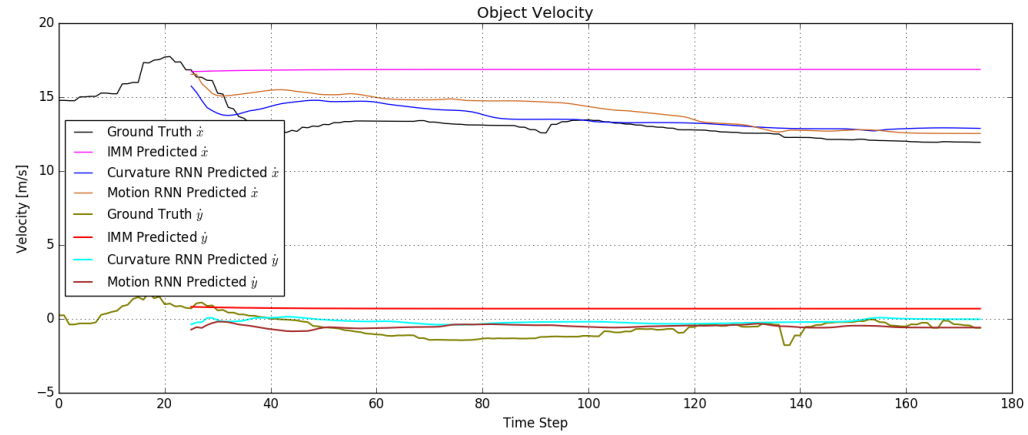


(c)

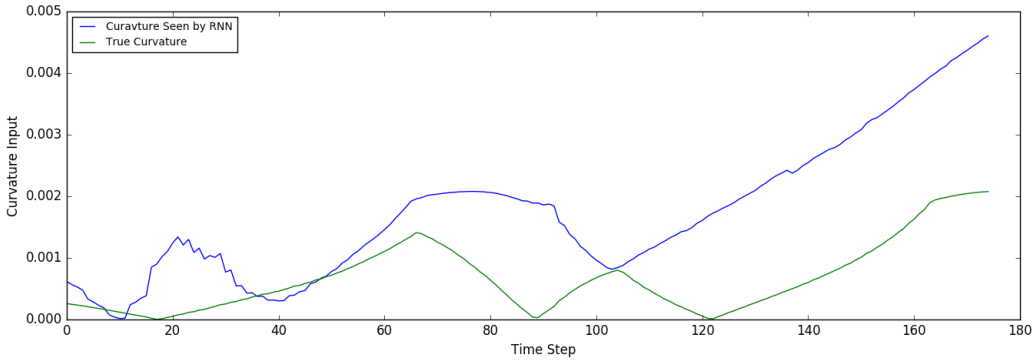
Figure A-12: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature



(a)

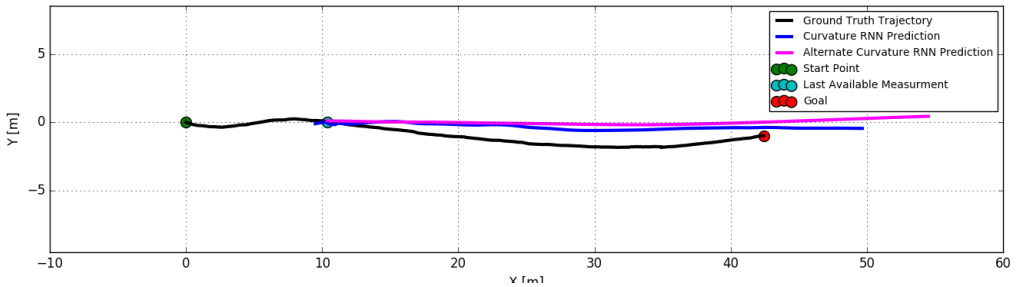


(b)

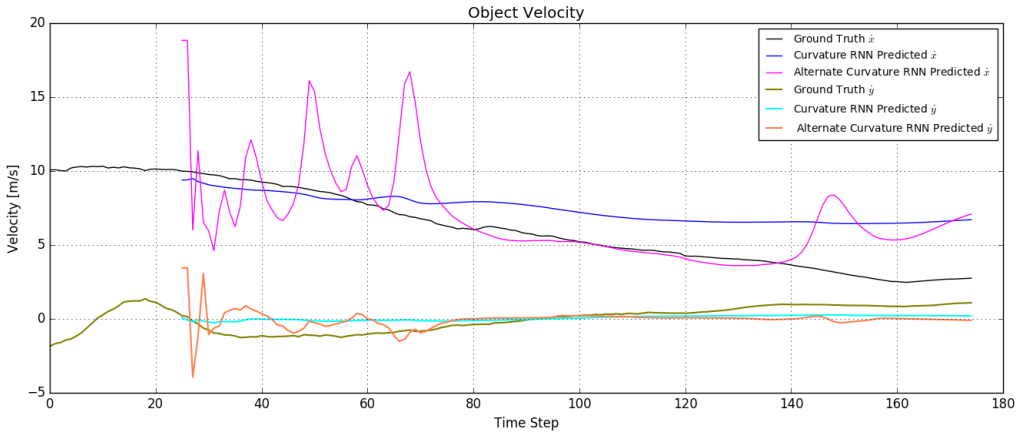


(c)

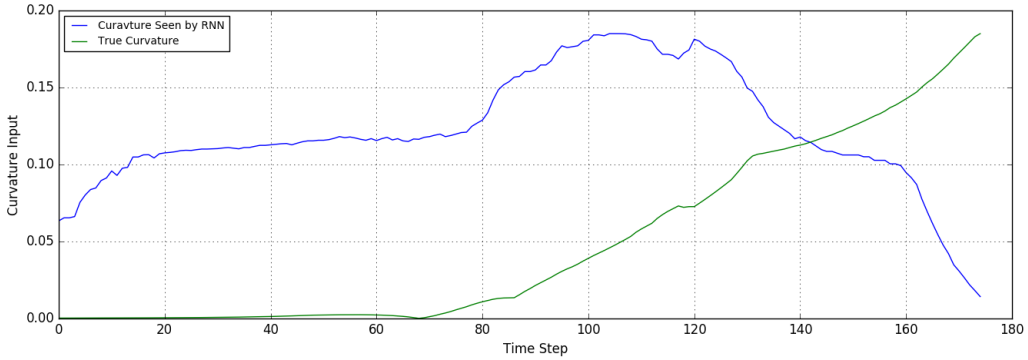
Figure A-13: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature



(a)

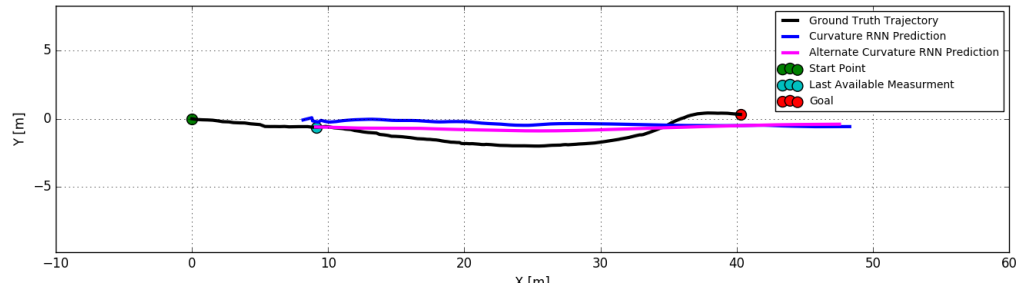


(b)

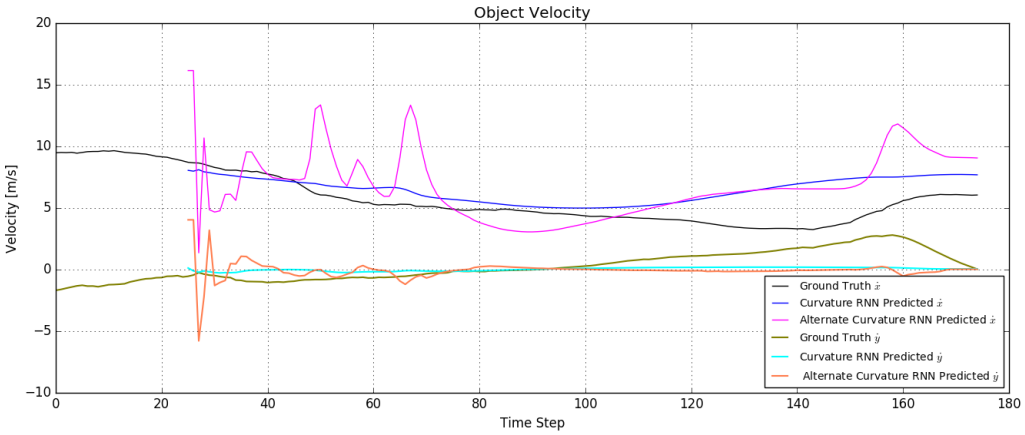


(c)

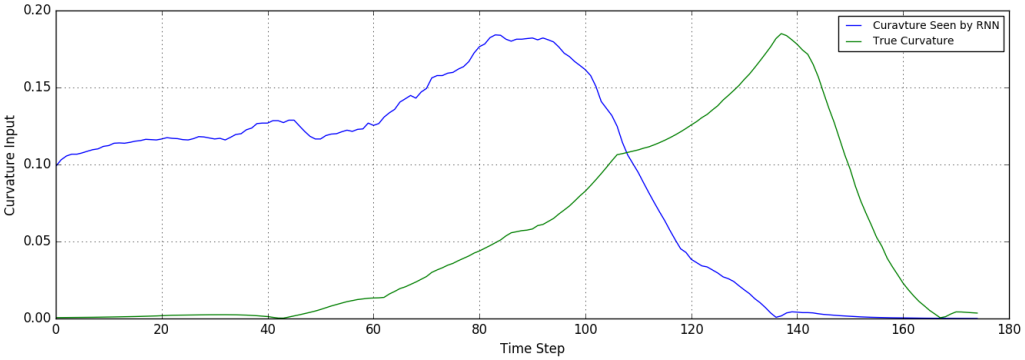
Figure A-14: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature



(a)

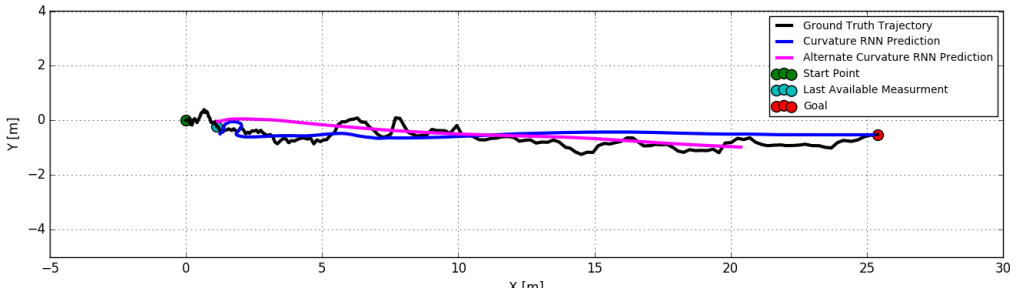


(b)

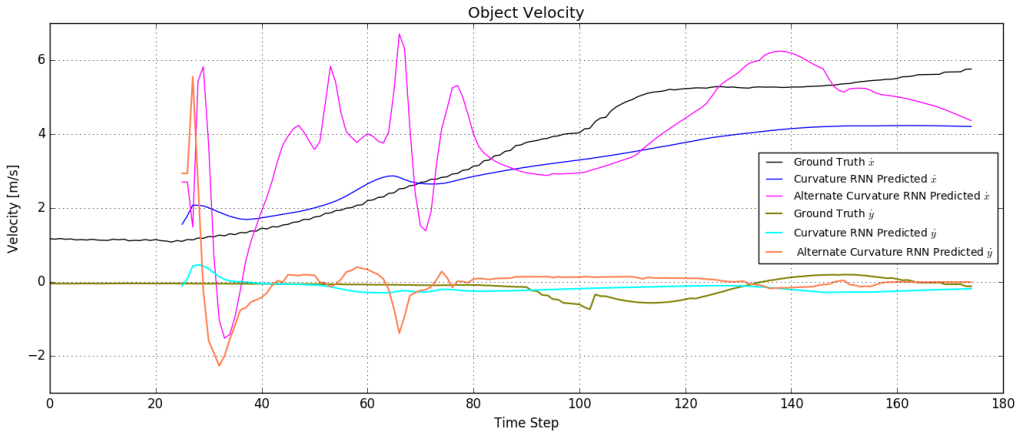


(c)

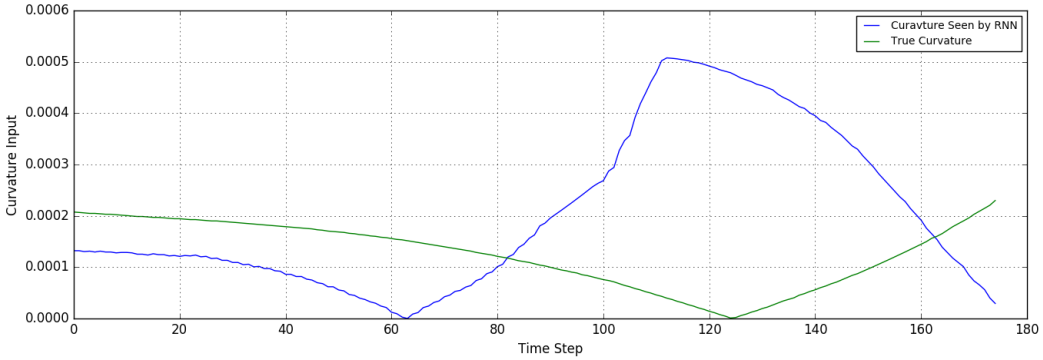
Figure A-15: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature



(a)



(b)



(c)

Figure A-16: Vehicle Trajectory predicted using IMM, Motion RNN and Curvature RNN models a) Vehicle Position plots, b) Vehicle Velocity plots, c) Road Curvature

Appendix B

Artificial Dataset

The performance of a neural network largely depends on the dataset that is used to train the model. For the neural network to generalize over the dataset and prevent overfitting, it is essential that the dataset must have a large number of samples. However, for this work, only a limited amount of trajectory data could be collected from the test vehicle. To address this, the Sequence-to-Sequence Recurrent Neural Network is trained under a transfer learning setting. The model is first trained on an artificial dataset and then on the dataset of recorded trajectories. We describe the artificial dataset in this chapter.

The artificial consists of trajectories of a maneuvering vehicle in the Curvilinear Coordinate System. The trajectories have motion only on the S axis, this representing the longitudinal vehicle motion. Gaussian noise is added two motion on the both axis based on covariances obtained from the LIDAR sensor used to collect real vehicle trajectories.

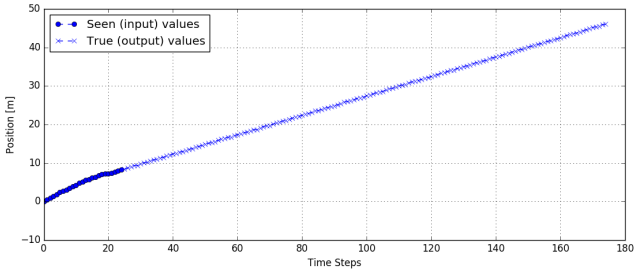
There are two types of trajectories represented in the artificial dataset, vehicle trajectories on straight roads and vehicle trajectories on curved roads section. We describe each of these separately in the following sections.

B-1 Vehicle Trajectories on straight roads

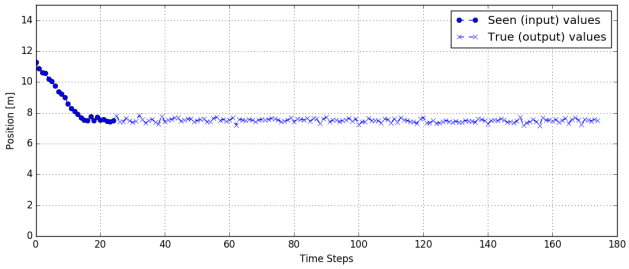
These trajectories represent different vehicle (longitudinal) maneuvers on a straight road, such as vehicle acceleration to a constant velocity, or breaking. Figure B-1, B-2, B-3 shows a few sample trajectories. Each trajectory is formed by a combination of the constant velocity or the constant acceleration motion, such that the duration and the sequence of both the motion types is chosen randomly. We assume maximum comfortable acceleration of a vehicle to be $2m/s$ [15]. The curvature value for all these trajectories is zero (inverse of infinity radius).

In each trajectory, the vehicle only maneuvers in the input sequence that is input to a prediction model, and has a linear behavior further on. This is done because the only feature we consider to cause a non-linearity in motion described in the CCS is road curvature. However,

these set of trajectories simulate a vehicle's motion on a straight road. As a result, a prediction model would not have any means to anticipate a maneuver such as accelerating to higher speed during prediction, and this would not lead to optimization of the network parameters.

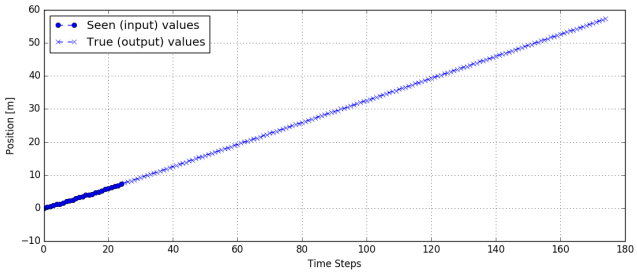


(a)

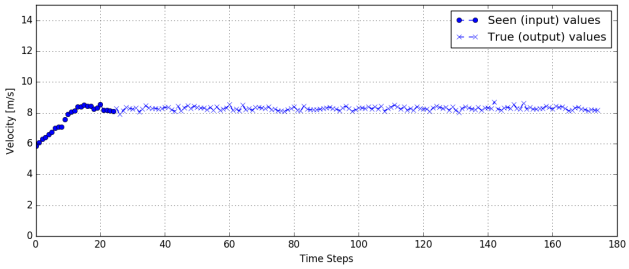


(b)

Figure B-1: Example trajectory for a vehicle motion on straight road, a) Vehicle position and b) Velocity on S axis of CCS

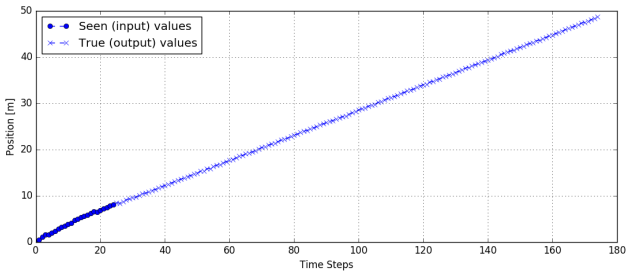


(a)

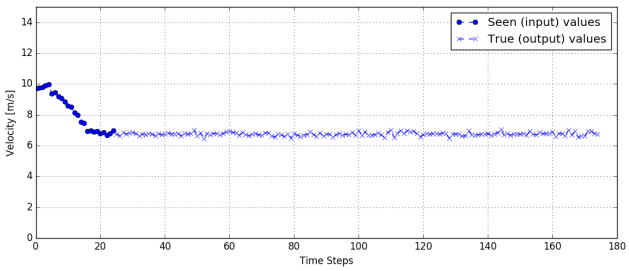


(b)

Figure B-2: Example trajectory for a vehicle motion on straight road, a) Vehicle position and b) Velocity on S axis of CCS



(a)



(b)

Figure B-3: Example trajectory for a vehicle motion on straight road, a) Vehicle position and b) Velocity on S axis of CCS

B-2 Vehicle Trajectories on curved roads

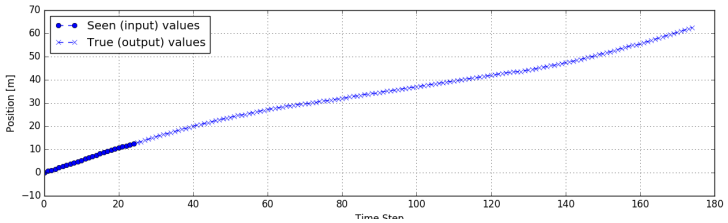
This set of trajectories represent the vehicle motion curved road sections of different radius. To simulate these vehicle trajectories, we find the maximum velocity v_{max} that a vehicle can safely travel with on a curve road using the following equation:

$$v = \sqrt{\frac{e * g}{k}} \tag{B-1}$$

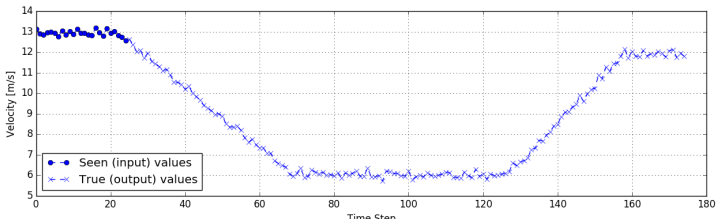
where μ is coefficient of friction, g is gravitational acceleration, and k is road curvature.

We randomly select an inital vehicle velocity as well as a velocity close to v_{max} as the intended velocity of the vehicle for negotiating the curve. Based on these velocities[1], a trajectory is obtained by combining constant velocity and constant acceleration motion type on the longitudinal axes. This is done for curves of four radii values, $12.5m, 15m, 20m, 25m$, each of which gives different v_{max} .

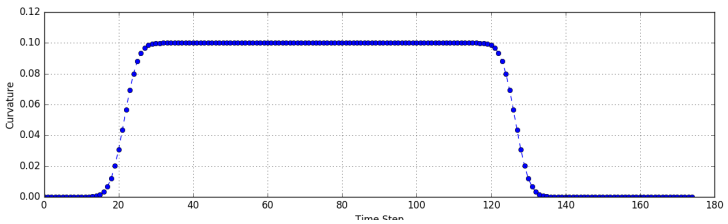
Figure B-4 gives a sample trajectory.



(a)



(b)



(c)

Figure B-4: Example trajectory for a vehicle motion on curved road, a) Vehicle position and b) Velocity on S axis of CCS

[1]We assume that a vehicle would decelerate to near V_{max} velocity prior to approaching the curve.

Bibliography

[1] K. Jo, M. Lee, J. Kim, and M. Sunwoo, "Tracking and behavior reasoning of moving vehicles based on roadway geometry constraints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, pp. 460–476, 2017.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[4] "Annual accident report 2016, the care database."

[5] R. Iden and S. A. Shappell, "A human error analysis of us fatal highway crashes 1990–2004," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 50, pp. 2000–2003, Sage Publications, 2006.

[6] M. Heuer, A. Al-Hamadi, M.-M. Meinecke, and R. Mende, "Requirements on automotive radar systems for enhanced pedestrian protection," in *Radar Symposium (IRS), 2012 13th International*, pp. 45–48, IEEE, 2012.

[7] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *Robomech Journal*, vol. 1, no. 1, p. 1, 2014.

[8] S. Lefèvre, J. Ibañez-Guzmán, and C. Laugier, "Context-based estimation of driver intent at road intersections," in *Computational Intelligence in Vehicles and Transportation Systems (CIVTS), 2011 IEEE Symposium on*, pp. 67–72, IEEE, 2011.

[9] M. Liebner, F. Klanner, M. Baumann, C. Ruhhammer, and C. Stiller, "Velocity-based driver intent inference at urban intersections in the presence of preceding vehicles," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 2, pp. 10–21, 2013.

[10] P. Liu, A. Kurt, *et al.*, "Trajectory prediction of a lane changing vehicle based on driver behavior estimation and classification," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 942–947, IEEE, 2014.

[11] Y. Hou, P. Edara, and C. Sun, "Situation assessment and decision making for lane change assistance using ensemble learning methods," *Expert Systems with Applications*, vol. 42, no. 8, pp. 3875–3882, 2015.

[12] C. Laugier, I. E. Paromtchik, M. Perrollaz, M. Yong, J.-D. Yoder, C. Tay, K. Mekhnacha, and A. Nègre, "Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 4, pp. 4–19, 2011.

[13] P. Dawkins, "Arc length with parametric equations."

[14] E. W. Weisstein, ""Curvature" from mathworld–a wolfram web resource."

[15] C. Gámez Serna and Y. Ruichek, "Dynamic speed adaptation for path tracking based on curvature information and speed limits," *Sensors*, vol. 17, no. 6, p. 1383, 2017.

[16] J. Hillenbrand, A. M. Spieker, and K. Kroschel, "A multilevel collision mitigation approachâĂŤits situation assessment, decision making, and performance tradeoffs," *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 4, pp. 528–540, 2006.

[17] R. Miller and Q. Huang, "An adaptive peer-to-peer collision warning system," in *Vehicular technology conference, 2002. VTC Spring 2002. IEEE 55th*, vol. 1, pp. 317–321, IEEE, 2002.

[18] B. Kim, C. M. Kang, S. H. Lee, H. Chae, J. Kim, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," *arXiv preprint arXiv:1704.07049*, 2017.

[19] S.-S. Jan and Y.-C. Kao, "Radar tracking with an interacting multiple model and probabilistic data association filter for civil aviation applications," *Sensors*, vol. 13, no. 5, pp. 6636–6650, 2013.

[20] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4694–4702, 2015.

[21] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1110–1118, 2015.

[22] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[23] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks.," in *AAAI*, pp. 4225–4232, 2017.

[24] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–971, 2016.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[26] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, pp. 1310–1318, 2013.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.