UNIVERSITY OF CALIFORNIA, SAN DIEGO

Using Deep Learning to Predict Obstacle Trajectories for Collision Avoidance in Autonomous Vehicles

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Jaskaran Virdi

Committee in charge:

Professor Henrik Christensen, Chair
Professor David Kreigman
Professor Hao Su

2018

The thesis of Jaskaran Virdi is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

Chair

University of California, San Diego

2018

DEDICATION

To my mother, Heman, who has always been a positive role model in my life. My mother has been a lifelong example of hard work and perseverance. She has not lived the easiest of lives, giving up her successful career to raise me and my brother, yet she is constantly pushing forward, kindly and hopefully, yet never backing down from her beliefs all the while. Thank you, mom, for your unconditional love and support through the difficult times.

EPIGRAPH

I have not failed. I just found 10,000 things that don't work.

—Thomas Edison

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thanks to Professor Henrik Christensen for allowing me to work in his lab on such a cutting edge problem and sharing his wisdom and experience on autonomous vehicles in this project. It wouldn't have been possible without his infectious enthusiasm and guidance which has also inspired me to work on challenging problems in the field of autonomous vehicles. I would also like to thank Professor David Kreigman and Professor Hao Su for taking out the time to be a part of the review committee for this thesis.

ABSTRACT OF THE THESIS

Using Deep Learning to Predict Obstacle Trajectories for Collision Avoidance in Autonomous Vehicles

by

Jaskaran Virdi

Master of Science in Computer Science

University of California, San Diego, 2018

Professor Henrik Christensen, Chair

As a part of developing autonomous vehicles and better Advanced driver assistance systems(ADAS), it is important to consider how the spatio-temporal activities of other agents in the environment like pedestrians, vehicles, etc. which are competing for space on roads might impact the motion planning performance of the vehicle . A system which can predict future obstacle trajectories as well as warn the driver or the autonomous vehicle about an impending collision will lead to safer roads and save lives.

Previous vehicle trajectory prediction approaches use motion models which have assumptions like constant velocity or constant acceleration which doesn't generalize well. Our approach

is completely data driven and gives promising results for predicting trajectory of the obstacle up to 2 seconds in the future using a deep recurrent neural network.

Taking inspiration from the recent success of sequence-to-sequence models in language translation we apply sequence-to-sequence recurrent neural networks to the new problem of trajectory prediction. The proposed scheme feeds the sequence of obstacles' past trajectory data obtained from sensors like LIDAR and GPS to the LSTM and predicts the position of the obstacle at future time steps. We use the KITTI dataset which provides us with annotated trajectory data for learning and evaluation.

Chapter 1

Introduction

Autonomous vehicles are an important area of research because of their benefits to transportation and our daily lives. On busy urban roads, the driving environment is changing all the time. For the safety of the autonomous vehicle and other traffic participants, the autonomous vehicle needs to estimate the future motion of the surrounding traffic participants and find the feasible moving region before planning its own trajectory. Therefore, trajectory prediction for participants constitutes the basis of trajectory planning, which is critical in achieving safe autonomous driving. For autonomous vehicles, the goal of trajectory prediction is to recognize the future trends of the relative motion between their surrounding obstacles and themselves.

In this work, we propose an application of the state of the art in sequence prediction, i.e. sequence-to-sequence[SVL14] recurrent neural networks, for solving the problem of future trajectory prediction which gives us promising results.

This section first describes about Advanced Driver Assistance Systems(ADAS) and how they have made driving safer. Then we describe the basic working of a collision avoidance system. After that we introduce the important subsystems in an autonomous vehicle and how they interact with each other. We then briefly explain the path planning phase in an autonomous vehicle and show how trajectory prediction is an important part in safe path planning.

1.1 Advanced Driver Assistance Systems

Advanced driver assistance systems (ADAS)[con17a] are systems developed to enhance vehicle systems for safety and better driving. Safety features are designed to avoid collisions and accidents by offering technologies that alert the driver to potential problems, or to avoid collisions by implementing safeguards and taking over control of the vehicle. Adaptive features may automate lighting, provide adaptive cruise control, automate braking, incorporate GPS/ traffic warnings, connect to smartphones, alert driver to other cars or dangers, keep the driver in the correct lane, or show what is in blind spots. ADAS relies on inputs from multiple data sources, including automotive imaging, lidar, radar, image processing, computer vision, and in-car networking.

Some common examples of ADAS are:

- Adaptive cruise control (ACC)
- Glare-free high beam and pixel light
- Adaptive light control: swiveling curve lights
- Anti-lock braking system
- Automatic parking
- Automotive navigation system with typically GPS and TMC for providing up-to-date traffic information.
- Automotive night vision
- Blind spot monitor
- Collision avoidance system
- Crosswind stabilization

- Cruise control
- Driver drowsiness detection
- Driver Monitoring System
- Electric vehicle warning sounds used in hybrids and plug-in electric vehicles
- Emergency driver assistant
- Forward Collision Warning
- Intersection assistant
- Hill descent control
- Intelligent speed adaptation or intelligent speed advice (ISA)
- Lane departure warning system
- Lane change assistance
- Night Vision
- Parking sensor
- Pedestrian protection system
- Rain sensor
- Surround View system
- Tire Pressure Monitoring
- Traffic sign recognition
- Turning assistant

- Vehicular communication systems
- Wrong-way driving warning

1.2 Collision Avoidance

A collision avoidance system[con17b] is an automobile safety system designed to reduce the severity of a collision. It uses radar (all-weather) and sometimes laser (LIDAR) and camera (employing image recognition) to detect an imminent crash. GPS sensors can detect fixed dangers such as approaching stop signs through a location database.

Once the detection is done, these systems either provide a warning to the driver when there is an imminent collision or take action autonomously without any driver input (by braking or steering or both). Collision avoidance by braking is appropriate at low vehicle speeds (e.g. below 50 km/h), while collision avoidance by steering is appropriate at higher vehicle speeds.

1.3 Autonomous Driving

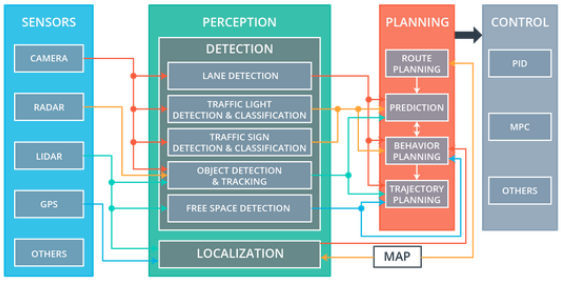

Figure 1.1: Autonomous Vehicle System

Figure 1.1 shows the important subsystems of an autonomous vehicle and how they coordinate with each other to accomplish autonomous driving. These subsystems are explained below[Sil17].

1.3.1 Sensors

An autonomous vehicle's sensor subsystem encompasses the physical hardware that gathers data about the environment. It usually consists of camera, RADAR, LIDAR, GPS and IMU.

Cameras are usually mounted behind the top of the windshield and are between 1-3 lined up in a row. These cameras are calibrated in stereo configuration and provide RGB video data. This raw RGB video data can then be processed to get a measure of scene depth based on stereo.

A RADAR(Radio Detection And Ranging) is an object-detection system that uses radio waves to determine the range, angle, or velocity of objects. It can be used to detect aircraft, ships, spacecraft, guided missiles, motor vehicles, weather formations, and terrain. A radar system consists of a transmitter producing electromagnetic waves in the radio or microwaves domain, a transmitting and receiving antenna, and a receiver and processor to determine properties of the object(s). Radio waves (pulsed or continuous) from the transmitter reflect off the object and return to the receiver, giving information about the object's location and speed. Radars can accurately detect and track objects as far as 200m away. Radars also perform extremely well in adverse weather conditions and are available at an affordable price point.

LIDAR (also called Light Imaging, Detection, And Ranging) is a surveying method that measures distance to a target by illuminating that target with a pulsed laser light, and measuring the reflected pulses with a sensor. Differences in laser return times and wavelengths can then be used to make digital 3D-representations of the target called point clouds. The detail captured in LIDAR is far greater than that of a RADAR.

IMU(Inertial Measurement Unit) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the magnetic field surrounding the body, using a combination of accelerometers and gyroscopes, sometimes also magnetometers. An IMU allows a GPS receiver to work when GPS-signals are unavailable, such as in tunnels, inside buildings, or when electronic interference is present. IMU uses the concept of dead reckoning, which is the

process of calculating one's current position by using a previously determined position, or fix, and advancing that position based upon known or estimated speeds over elapsed time and course. The guidance system is continually integrating acceleration with respect to time to calculate velocity and position, any measurement errors, however small, are accumulated over time which causes drift. This drift is corrected by using a GPS in tandem with an IMU. This is done in automotive navigation systems or vehicle tracking systems, giving the system a dead reckoning capability and the ability to gather as much accurate data as possible about the vehicle's current speed, turn rate, heading, inclination and acceleration, in combination with the vehicle's wheel speed sensor output and, if available, reverse gear signal, for purposes such as better traffic collision analysis.

1.3.2 Perception

The perception subsystem translates raw sensor data into meaningful intelligence about the autonomous vehicle's environment. The components of the perception subsystem can be grouped into two blocks: detection and localization.

The detection block uses sensor information to detect objects outside the vehicle. These detection components include traffic light detection and classification, object detection and tracking, and free space detection. The localization block determines where the vehicle is in the world. GPS is only accurate to within 1 to 2 meters. For a car, such an error range is unacceptably large. A car that thinks its in the center of a lane could be off by a meter and really be on the sidewalk, running into things.

For localization, usually SLAM(Simultaneous Localization and Mapping) is performed on data obtained from various sources like LIDAR, RADAR and Cameras.

1.3.3 Planning

A planner builds a series of waypoints for the autonomous vehicle to follow. These waypoints are just spots on the road that the autonomous vehicle needs to drive over. Each waypoint has a specific location and associated target velocity that the autonomous vehicle should match when it passes through that waypoint. The planner uses the perception data to predict the movements of other vehicles on the road and update the waypoints accordingly.

For example, if the planning subsystem were to predict that the vehicle in front would be slowing down, then the planner would likely decide to decelerate the autonomous vehicle.

Prediction of trajectory of obstacles such as cars, buses, trucks, pedestrians, etc. is a very important part of the planning subsystem in an autonomous vehicle. This prediction plays an important role for the planner to decide the optimal trajectory for the autonomous vehicle making path planning safer.

1.3.4 Control

The control subsystem actuates the vehicle by sending acceleration, brake, and steering messages. Some of these messages are purely electronic, and others have a physical manifestation. For example, turning of steering wheel in an autonomous vehicle. The control subsystem takes as input the list of waypoints and target velocities generated by the planning subsystem. Then the control subsystem passes these waypoints and velocities to an algorithm such as a PID controller, which calculates just how much to steer, accelerate, or brake, in order to hit the target trajectory.

Chapter 2

Background

We first talk about the some of the foundational work done in the field of trajectory prediction for vehicles. Then we motivate the use of recurrent neural networks for trajectory prediction. We then follow this up with a short description of recurrent neural networks and their improved variants. In the end of this chapter, we describe about sequence-to-sequence model which is the state of the art in terms of sequence prediction problems and motivate the use of encoder-decoder LSTM network for real world trajectory prediction which is essentially a sequence prediction problem.

2.1 Related Work

Previous vehicle trajectory approaches use motion models[SRW08] such as Constant Velocity(CV) or Constant Acceleration(CA) which are linear in nature. These models assume straight motions and aren't able to take into account the yaw rate. Curvilinear models such as Constant Turn Rate and Velocity (CTRV) model and Constant Turn Rate and Acceleration(CTRA) model take into account rotations around z-axis and are more complex than the linear models but assume that there is no correlation between velocity and yaw rate.

These motion models along with Kalman Filter[Kal60] are used to predict the trajectory

of the vehicle. The Kalman Filter doesn't work when system equations aren't linear and the noise doesn't follow the Additive White Gaussian Noise(AWGN) model.

Non-linear version of Kalman Filters such as the Extended Kalman Filter(EKF) and the Unscented Kalman Filter(UKF) are able to handle non-linearity only upto a certain extent. EKF[Lju79] is an approximate filter for nonlinear systems, based on first-order approximation. UKF[WVDM00] isn't able to handle extremely non-Gaussian noise.

Kalman Filters don't work well in multi-step prediction problems whereas predicting trajectory of an obstacle at multiple time steps is a multi-step prediction problem. According to [DH92], Recurrent Neural Networks perform significantly better than Kalman Filters in all situations especially, where Kalman assumptions are violated. It is reasonable to assume that a sufficiently large recurrent neural network would be capable of approximating the equations of a Kalman Filter.

In [WZY17], the authors predict the trajectory of the object using recurrent neural network. However, in this case the trajectory prediction is done in image coordinates for the purpose of visual tracking and is different from our approach since we predict the real-world trajectory of the obstacle.

In [KKL+17], the authors use a recurrent neural network for real-world obstacle trajectory prediction. However, they use multiple single-step LSTM models, one for each future time step corresponding to 0.5s, 1.0s, 2.0s in the future. This is different from our approach which uses an encoder-decoder LSTM for multi-step prediction. This is better than the single step approach since we don't need to train multiple models corresponding to each time step.

2.2 Recurrent Neural Network

Recurrent neural networks have connections that have loops, adding feedback and memory to the networks over time. This memory allows this type of network to learn and generalize across

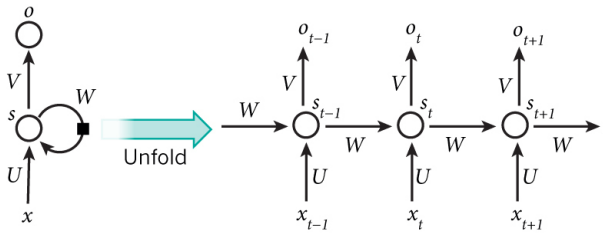

Figure 2.1: Recurrent Neural Network[LBH15]

sequences of inputs rather than individual patterns.

Below are the equations for RNN.

$$s_t = tanh(Ux_t + Ws_{t-1}) \tag{2.1}$$

$$o_t = softmax(V_t) \tag{2.2}$$

where

- s_t is the hidden state at time t
- x_t is the input at time t
- U is the weight matrix for input-hidden connection
- W is the weight matrix for the hidden-hidden connection
- V is the weight matrix for hidden-output connection
- o_t is the output at time t

Recurrent Neural Networks are trained using BPTT (Backpropagation Through Time) algorithm. They suffer from the problem of vanishing or exploding gradients when BPTT is applied for a large number of time steps. This problem of vanishing and exploding gradients over

a large number of time steps has been alleviated by introducing a new type of architecture called the Long Short-Term Memory Network.

2.3 Long Short Term Memory

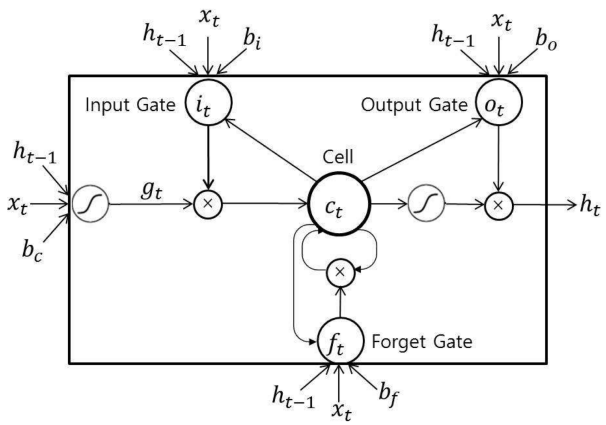

Figure 2.2: LSTM cell structure

The Long Short-Term Memory[HS97], the output, and the previous state of the cell. Unlike recurrent neural network, LSTM has a gating control mechanism that allows the network to forget[GSC99] past state in the memory or learn when to update its state given new information. Let c_t be the state of the memory cell at the current time step t. Then, c_t is updated by the following recursive equations

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{2.3}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{2.4}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{2.5}$$

$$g_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.7}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.8}$$

where

- $\sigma(x) = \frac{1}{1+e^{-x}}$: sigmoid function
- $x \odot y$: element-wise product
- $W_{xi}, W_{hi}, W_{xf}, W_{hf}, W_{xo}, W_{ho}, W_{xc}, W_{hc}$: weight matrix for linear transformation
- b_i, b_f, b_o, b_c: bias vector
- i_t: input gating vector
- f_t: forget gating vector
- o_t: output gating vector
- g_t: state update vector
- h_t: output hidden state vector

The input gate i_t and the output gate f_t can control the information flow from the input and to the output, respectively. The behavior of the gate control is learned from data as well. When unfolded in time, the LSTM is considered to be a deep model. We can also make the cell to have deep architecture by adding additional layers into the cell.

2.4 Sequence Prediction using Recurrent Neural Network

There are four primary models for sequence prediction. The following terminology is used in this section:

- X: The input sequence value, may be delimited by a time step, e.g. X(1).
- u: The hidden state value, may be delimited by a time step, e.g. u(1).
- y: The output sequence value, may be delimited by a time step, e.g. y(1)

2.4.1 One-to-One Model

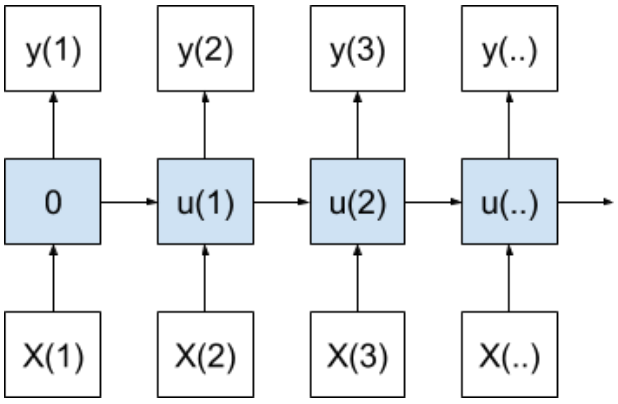

Figure 2.3: One-to-One Sequence Prediction Model Over Time

A one-to-one model produces one output value for each input value. The internal state for the first time step is zero; from that point onward, the internal state is accumulated over the prior time steps. In the case of a sequence prediction, this model would produce one time step forecast for each observed time step received as input. This is a poor use for RNNs as the model has no chance to learn over input or output time steps

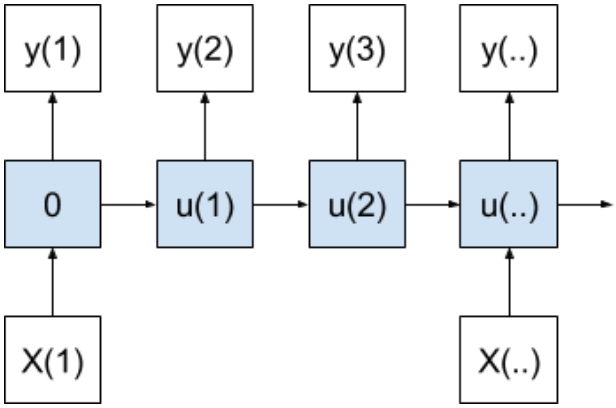

Figure 2.4: One-to-Many Sequence Prediction Model Over Time

2.4.2 One-to-Many Model

A one-to-many model produces multiple output values for one input value. This model can be used for image captioning where one image is provided as input and a sequence of words are generated as output.

2.4.3 Many-to-One Model

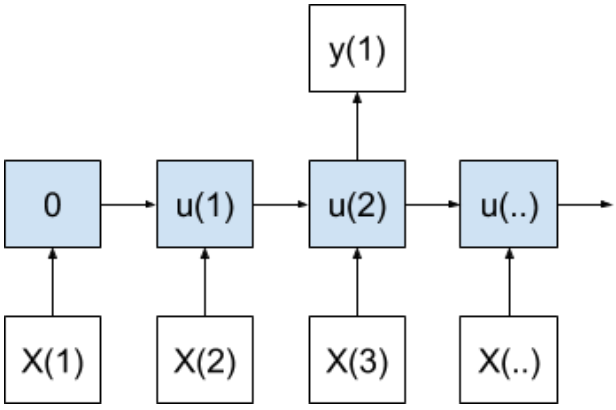

Figure 2.5: Many-to-One Sequence Prediction Model Over Time[Bro17]

A many-to-one model produces one output value after receiving multiple input values. The internal state is accumulated with each input value before a final output value is produced. This can be applied to video activity classification where a video frame is provided as input at

each time step and the output is the type of activity being done in the video.

2.4.4 Many-to-Many Model

A many-to-many model produces multiple outputs after receiving multiple input values. This model is used in language translation and is the basis of sequence-to-sequence learning

2.5 Encoder-Decoder Architecture

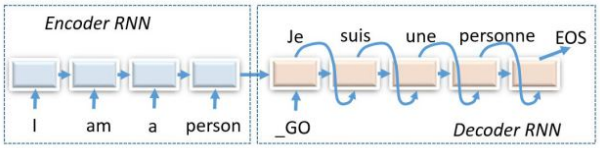

Figure 2.6: Encoder Decoder RNN

The Encoder-Decoder LSTM is a recurrent neural network designed to address sequence-to-sequence problems, sometimes called seq2seq[SVL14]. Sequence-to-sequence prediction problems are challenging because the number of items in the input and output sequences can vary. For example, text translation is an example of seq2seq problems. This problem is an example of many-to-many type sequence prediction problem.

The encoder-decoder architecture for solving sequence-to-sequence problems comprises of two models: one for reading the input sequence and encoding it into a fixed-length vector, and a second for decoding the fixed-length vector and outputting the predicted sequence. The Encoder-Decoder LSTM was developed for natural language processing problems where it demonstrated state-of-the-art performance, specifically in the area of text translation called statistical machine translation.

The innovation of this architecture is the use of a fixed-sized internal representation in the heart of the model that input sequences are read to and output sequences are read from. For this

reason, the method may be referred to as sequence embedding.

In one of the first applications of the architecture to English-to-French translation, the internal representation of the encoded English phrases was visualized. The plots revealed a qualitatively meaningful learned structure of the phrases harnessed for the translation task.

Chapter 3

Methodology

In this section, we first describe the properties of the KITTI raw data and why we chose it for evaluation. Then, we describe the input data to our deep learning model and how we acquired it from the dataset. After that we compare and contrast the architecture of the deep learning models.

3.1 The KITTI Raw Data

The KITTI[GLSU13] dataset is one of the most popular benchmark datasets in autonomous driving. This dataset has a wide range of scenarios that are diverse, capturing real-world traffic situations, and range from freeways over rural areas to inner-city scenes with many static and dynamic objects. Another motivation to use this dataset was that it provides us with annotated 3D bounding boxes corresponding to all the dynamic obstacles within the the ego-car camera's field of view. This is the ground-truth which we use in evaluating the correctness of our trajectory predictions. This is helpful to us because our work focuses on obstacle trajectory prediction rather than obstacle trajectory tracking.

The raw data is divided into the categories 'Road', 'City', 'Residential', 'Campus' and 'Person'.The dataset provides recorded sensor information from camera, LiDAR and GPS/IMU

tags obtained from driving a Volkswagen Passat B6 around the city of Karlsruhe in Germany, in rural areas and highways.

The dataset comprises the following information, captured and synchronized at 10 Hz:

- Raw (unsynced+unrectified) and processed (synced+rectified) grayscale stereo sequences (0.5 Megapixels, stored in png format)
- Raw (unsynced+unrectified) and processed (synced+rectified) color stereo sequences (0.5 Megapixels, stored in png format)
- 3D Velodyne point clouds (100k points per frame, stored as binary float matrix)
- 3D GPS/IMU data (location, speed, acceleration, meta information, stored as text file)
- Calibration (Camera, Camera-to-GPS/IMU, Camera-to-Velodyne, stored as text file)
- 3D object tracklet labels (cars, trucks, trams, pedestrians, cyclists, stored as xml file). The tracklet labels file includes information about all the tracklets in every frame of the video sequence obtained from the cameras. For each frame and for each tracklet in that frame, one can get the coordinates of the eight corners of bounding box for that tracklet. The coordinates of the bounding box can be obtained in the velodyne coordinate system which is a frame of reference mounted to the car.

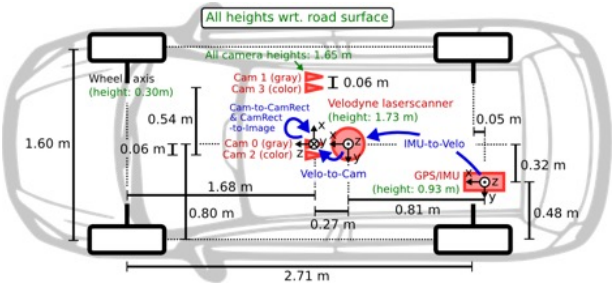

Figure 3.1: Coordinate systems used in the KITTI dataset.

A total of 6383 video frames of annotated data was used corresponding to all the annotated data from the Road and City categories. Corresponding to each video frame we have bounding

box vertex coordinates in Velodyne laser scanner coordinate frame for all the obstacles present in that frame.

3.2 Features from Trajectory Data

The features which we use in our models are measured in the Velodyne laser scanner's frame which is fixed to the ego-car. Thus, the measurements are made from a moving frame and thus, one needs ego-car's features for ego-motion compensation.

The input sequence length is of 20 corresponding to the history of 2 seconds sampled at 10 Hz. For every sample input sequence to LSTM, a total of 9 features are fed at each time step to the LSTM models. These features are:

3.2.1 Ego-Vehicle Features

The below features are extracted using GPS/IMU tags

- Yaw Rate i.e angular rate about Velodyne laser scanner's z-axis (rad/s)
- Forward Velocity, i.e. parallel to earth-surface (m/s))
- Leftward Velocity, i.e. parallel to earth-surface (m/s))
- Forward Acceleration(m/s^2)
- Leftward Acceleration(m/s^2)

3.2.2 Obstacle Features

- Obstacle bounding box center coordinates. From the Figure 3.1, one can see that in the Velodyne laser scanner coordinate system, the x coordinate refers to the longitudinal direction i.e positive in the direction of the ego-vehicle and negative in the opposite direction.

The y coordinate refers to the lateral direction i.e positive in left and negative in right. Both coordinates are measured in m.

- Obstacle bounding box center velocity in both the longitudinal and lateral directions is measured in the Velodyne laser scanner's coordinate frame in m/s.

3.3 Machine Learning Models

This is a supervised regression sequence learning problem. All models below use the above features from the past 2 seconds to predict future trajectory of up to 2 seconds.

3.3.1 Baseline

The baseline model is linear regression which regresses only on the past trajectory data(bounding box center coordinates) of 2 seconds to predict future trajectory.

3.3.2 Single Step Path prediction using LSTM

In this case, an LSTM model is trained for each forecast horizon. So, there are different LSTM models corresponding to each of the forecast horizons of 0.5 seconds, 1 seconds, 1.5 seconds and 2 seconds. Each LSTM is a single step prediction model i.e the output is (x,y) for a single time instant in the future for the obstacle. This model is our implementation of what was used in [KKL+17].

3.3.3 Multiple Step Path prediction using Encoder-Decoder LSTM Architecture

The encoder-decoder model uses the above features from past trajectory for the obstacle for 2 seconds and then outputs the coordinates of the future trajectory after every 0.1s(since data

is sampled at 10Hz) till 2 seconds. Thus, in this case the output is a sequence of length 20 time steps and each time step consists of the obstacle's bounding box center coordinates(x,y). This is different from the previous non-baseline model which only gives prediction at a single time step in the future.

Chapter 4

Results

Table 4.1: Future Trajectory Prediction Error (MAE in m) for KITTI dataset. For each model, first column is error in X(longitudinal) and second column is error in Y(lateral)

Future Time(secs)	Baseline		Single-step LSTM		Encoder-Decoder LSTM	
0.5	0.71 ± 1.09	0.55 ± 1.16	$\mathbf{0.16 \pm 0.20}$	$\mathbf{0.13 \pm 0.25}$	0.23 ± 0.58	0.20 ± 0.31
1.0	1.40 ± 1.96	0.97 ± 2.01	0.58 ± 1.79	0.40 ± 1.11	$\mathbf{0.51 \pm 0.71}$	$\mathbf{0.45 \pm 0.72}$
1.5	2.22 ± 3.00	1.41 ± 3.03	$\mathbf{0.94 \pm 0.93}$	$\mathbf{0.75 \pm 1.11}$	0.98 ± 1.48	0.81 ± 1.91
2.0	3.24 ± 4.15	1.85 ± 4.00	1.71 ± 3.30	1.22 ± 1.98	$\mathbf{1.55 \pm 1.77}$	$\mathbf{1.18 \pm 2.16}$

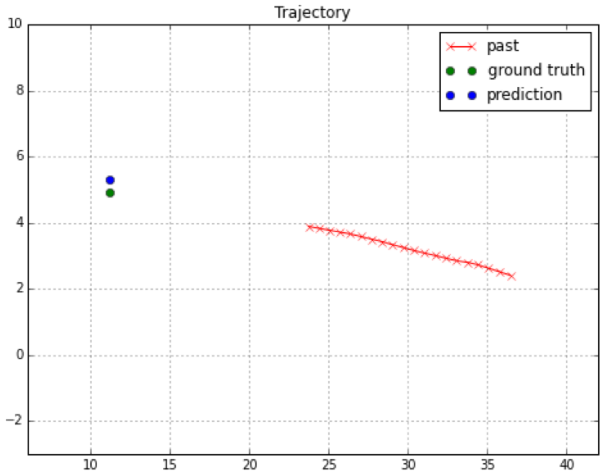

Figure 4.1: Single-step model future path prediction for 2 seconds for a straight trajectory

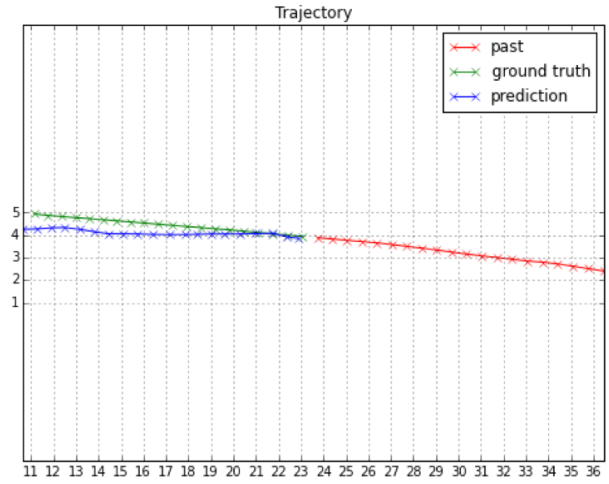

Figure 4.2: Encoder-Decoder model future path prediction for 2 seconds for a straight trajectory

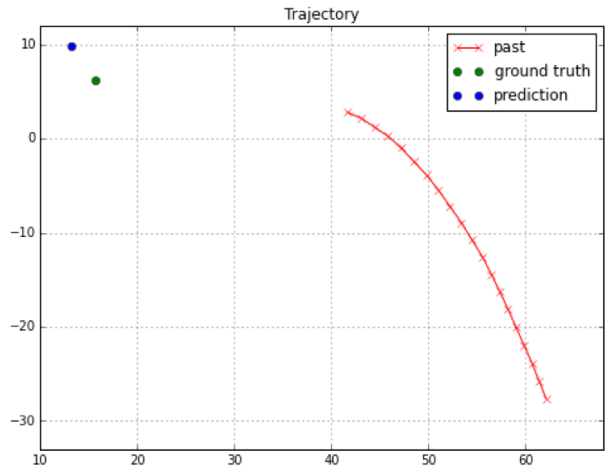

Figure 4.3: Single-step model future path prediction for 2 seconds for a curved trajectory

4.1 Evaluation Metric

$$MAE = \frac{\sum_{i=1}^{n} |y_i - pred_i|}{n} \tag{4.1}$$

We use the evaluation metric of Mean Absolute Error(MAE) to report the error between the ground truth(y_i) and prediction($pred_i$) measured(in the Velodyne laser scanner coordinate system attached to the ego-car) at multiple time steps in both the longitudinal direction(X) and

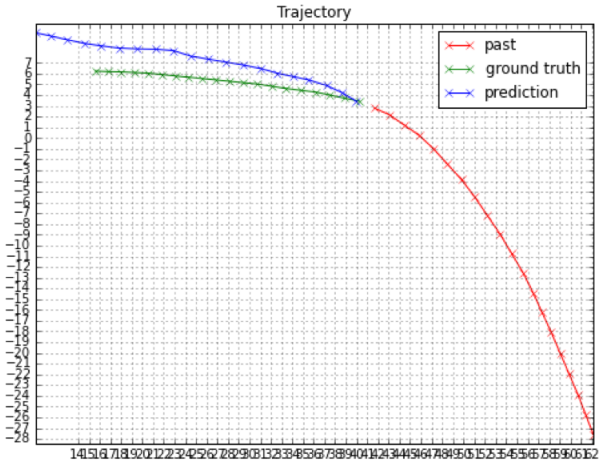

Figure 4.4: Encoder-Decoder model future path prediction for 2 seconds for a curved trajectory

lateral direction(Y).

4.2 Analysis

From Table 4.1, we see that both the deep learning models beat the baseline by a strong margin. Our encoder-decoder LSTM method gives promising results for path prediction and has similar performance to the model used in [KKL+17]. The advantage of our model is that one doesn't need to train multiple models for different time steps. In our case, we just train one model which can predict the future trajectory at multiple time steps. We show results of the single-step and multi-step LSTM models on two different trajectories in the dataset. One of them is straight and other is curved. For the straight trajectory, Figure 4.1 and Figure 4.2 show results of single step LSTM and encoder-decoder LSTM respectively. For the curved trajectory, Figure 4.3 and Figure 4.4 show results of single step LSTM and encoder-decoder LSTM respectively. For all the trajectory plots, each point on the trajectory represents the position of the obstacle after every 0.1 second.

4.3 Learning Details

We use Keras[C+15] and TensorFlow[AAB+15] to write our deep learning models and Matplotlib [Hun07] for graphs. We use the mean squared error loss for training and train both the deep learning models with Adam[KB14] optimizer using initial learning rate of 0.004 for 600 epochs on the annotated data corresponding to 6383 video frames from both the Road and City scenes of KITTI dataset. All models are trained end-to-end with a NVIDIA Pascal GPU. We use early stopping to stop the training process if the mean absolute error increases on the validation test set. The patience for early stopping is set to 100 epochs. We apply gradient clipping with L2 norm of 1.0. The learning rate is reduced by a factor of 0.5 if the validation mean absolute error stops decreasing with a patience of 5 epochs and minimum learning rate allowed is 0.0001

The split of obstacle(cars, trucks, trams, pedestrians, cyclists) track data in training, validation and test is 75-5-20 respectively. History window used is of 2 seconds and is fixed in all cases. Sample rate is 10 Hz.

Only those obstacle tracks which have a minimum observed duration equal to the sum of history window and forecast horizon are used. In the case of 0.5 seconds forecast horizon, the number of such obstacle tracks are 379. For the 1.0 seconds forecast horizon, number of such obstacle tracks are 309. In the case of 1.5 seconds forecast horizon, number of such obstacle tracks are 237. For the case of 2.0 seconds forecast horizon, the number of such obstacle tracks are 186. There is a decrease in obstacle tracks as the forecast horizon increases because less number of tracks satisfy the above constraint as the forecast horizon increases.

Chapter 5

Discussion

We see that applying sequence to sequence learning using LSTM is a promising approach for predicting future obstacle trajectory. Our approach doesn't have any assumptions on either the motion model(constant velocity or constant acceleration) or a noise model(Gaussian noise) and is able to learn all the motion parameters from the trajectory data alone.

From the results, we see that with an increase in the forecast horizon, the error in future prediction increases drastically.

The increase in error over longer forecast horizons comes from the fact that the obstacle may start accelerating or decelerating at a different rate as compared to what was learned from its past trajectory. A classic example is sudden and unexpected breaking of a vehicle. Such trajectories are impossible to predict and leads to higher error.

The KITTI dataset which we used in our analysis is more challenging as compared to the dataset used in [KKL$^+$17] because we use data from both highway and city driving whereas [KKL$^+$17] their dataset consists of only highway driving where traffic is a lot more predictable eg: Only lane changes in highway driving as compared to stopping to a complete halt at a traffic light or sharp turns in city driving. The number of data points in KITTI dataset is a lot less as compared to [KKL$^+$17] which makes the learning problem all the more challenging.

Chapter 6

Conclusion

We introduce a new application of sequence-to-sequence LSTM model for predicting future trajectory of obstacles in an autonomous driving scenario which gives us promising results for prediction up to 2 seconds in the future on the challenging KITTI dataset.

Since, our work focuses on obstacle trajectory prediction rather than obstacle trajectory tracking, we used the KITTI dataset[GLSU13] which provides us with annotated track data. This data can be used as input to our deep learning model to give future path prediction. However, our system doesn't need human-in-the-loop for annotating obstacle track data since we are using the coordinates of the obstacle obtained after the forecast horizon as the label. Thus, our system has automatic logging and supports online system that can be trained during driving similar to [KKL$^+$17]. The coordinates and features of obstacle tracks can be obtained by sensor fusion of data from different sensor streams like camera, lidar and radar sensor and using existing vehicle detection and tracking algorithms such as Multiple Hypothesis Tracking(MHT).

This future path prediction system can be used either in ADAS as a collision avoidance system or in the path planning system of an autonomous car for safer trajectory planning. Thus, our path prediction system isn't specific to autonomous vehicles and can be used even on existing vehicles to make them safer as long as one has the above mentioned sensors to acquire data.

Chapter 7

Future Work

In this work, we predict the future trajectory of the obstacle by predicting the center coordinates of the bounding box annotation of the obstacle. However, this doesn't take into account the orientation of the obstacle. We can extend the encoder-decoder LSTM method to predict orientation of the obstacle in addition to the center coordinates of the obstacle at every time step.

We would also like to investigate the use of an obstacle tracker such as Multi-Hypothesis Tracker(MHT) instead of manually annotated trajectory data and how it affects the performance of prediction. The system would then consist of both obstacle tracking as well as their trajectory prediction without any dependence on manual work.

In our work, we use past trajectory data of 2 seconds with the assumption that the obstacle doesn't get occluded during that time. Thus, we would also like to explore occlusion handling in our system.

Another direction where we would like to explore would be in knowing the difficulty of trajectory prediction based on the type of the obstacle and if methods based on the type of obstacle are needed for trajectory prediction.

We would like to use visual cues such as image features of the obstacle for trajectory

prediction. This will take into account the appearance of the obstacle which can help in knowing the orientation of that obstacle. For e.g: If the obstacle is a car and it has its tail lights facing the ego-car there is a high chance that this obstacle is moving in the direction of ego-vehicle.

Incorporating information about the road map in the deep network is another way by which we can use visual cues to improve the trajectory prediction further.

Further analysis can be done on using different durations for past trajectory history and how it affects the future trajectory prediction.

We are also exploring the possibility of taking into account the static and dynamic scene context around the obstacle whose future path is predicted along with a need for a stochastic output which should give multiple possible predictions at each time step to improve the current prediction system further.

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[Bro17] Jason Brownlee, 2017. https://machinelearningmastery.com/models-sequence-prediction-recurrent-neural-networks/.

[C+15] François Chollet et al. Keras, 2015.

[con17a] Wikipedia contributors. Advanced driver-assistance systems — wikipedia, the free encyclopedia, 2017. [Online; accessed 17-December-2017].

[con17b] Wikipedia contributors. Collision avoidance system — wikipedia, the free encyclopedia, 2017. [Online; accessed 17-December-2017].

[DH92] J. P. DeCruyenaere and H. M. Hafez. A comparison between kalman filters and recurrent neural networks. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 247–251 vol.4, Jun 1992.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[GSC99] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[KKL+17] Byeoungdo Kim, Chang Mook Kang, Seung-Hi Lee, Hyunmin Chae, Jaekyum Kim, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. *CoRR*, abs/1704.07049, 2017.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[Lju79] Lennart Ljung. Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems. *IEEE Transactions on Automatic Control*, 24(1):36–50, 1979.

[Sil17] David Silver. How the udacity self-driving car works, 2017. https://medium.com/udacity/how-the-udacity-self-driving-car-works-575365270a40.

[SRW08] Robin Schubert, Eric Richter, and Gerd Wanielik. Comparison and evaluation of advanced motion models for vehicle tracking. In *Information Fusion, 2008 11th International Conference on*, pages 1–6. IEEE, 2008.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[WVDM00] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. Ieee, 2000.

[WZY17] Lituan Wang, Lei Zhang, and Zhang Yi. Trajectory predictor by using recurrent neural networks in visual tracking. *IEEE Transactions on Cybernetics*, 2017.