Socially and Spatially Aware Motion Prediction of Dynamic Objects for Autonomous Driving

by

Neel P. Bhatt

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2023

© Neel P. Bhatt 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Yang Shi Professor, Dept. of Mechanical Engineering University of Victoria
Supervisors:	Amir Khajepour Professor, Dept. of Mechanical and Mechatronics Engineering University of Waterloo
	Ehsan Hashemi Assistant Professor, Dept. of Mechanical Engineering University of Alberta Adjunct Assistant Professor, Dept. of Mechanical and Mechatronics Engineering, University of Waterloo
Internal Members:	Soo Jeon Associate Professor, Dept. of Mechanical and Mechatronics Engineering, University of Waterloo
	Arash Arami Assistant Professor, Dept. of Mechanical and Mechatronics Engineering, University of Waterloo
Internal-External Member:	Stephen L. Smith Professor, Dept. of Electrical and Computer Engineering University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-first-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The following first-author papers have appeared for publication and comprise the technical content of this thesis:

1. Neel P. Bhatt, Amir Khajepour, and Ehsan Hashemi. MPC-PF: Socially and Spatially Aware Object Trajectory Prediction for Autonomous Driving Systems Using Potential Fields. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 2023. [1]

2. Neel P. Bhatt, Amir Khajepour, and Ehsan Hashemi. MPC-PF: Social Interaction Aware Trajectory Prediction of Dynamic Objects for Autonomous Driving Using Potential Fields. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9837–9844, 2022. [2]

3. Ehsan Mohammadbagher, Neel P. Bhatt, Ehsan Hashemi, Baris Fidan, and Amir Khajepour. Real-time Pedestrian Localization and State Estimation Using Moving Horizon Estimation. In *23rd Intelligent Transportation Systems Conference (ITSC)*, 2020. [3]

Chapter 3 and Section 6.1 of this thesis contains material that has been presented in a publication that was co-first authored by myself, Mr. Ehsan Mohammadbagher, a PhD candidate, and my supervisors: Prof. Amir Khajepour and Prof. Ehsan Hashemi. I was the co-first author responsible for conceptualizing the research directions, laying out the research questions and hypotheses, designing and performing the experimental and theoretical analyses, and writing the draft manuscripts. Both myself and Mr. Mohammadbagher contributed equally to the work. My supervisors provided feedback and general supervision of the research. I am the sole author of all other thesis contents.

Abstract

The primary goal of this thesis project is to develop a robust object motion prediction framework enabling safe decision making for autonomous vehicles in various driving scenarios. Given the comparatively higher importance and complexity of urban driving settings such as stop-sign controlled intersections or non-signalized/sign controlled roads are of primary interest; the approach, however, is not limited to these settings and is applicable to other driving settings. Specifically, motion prediction for all moving objects surrounding the autonomous vehicle such as pedestrians, cyclists, cars, and trucks is considered.

In this thesis, estimation of the position and velocity of the objects surrounding the autonomous vehicle is performed using observed positions of the object in interest during a finite time window in the past, subsequent to which a socially and spatially informed model predicts the positions of these objects for a finite time window in the future through the use of the obtained position and velocity estimate as well as an artificial potential field (PF) modelling social interactions between surrounding objects and the scene.

The necessary inputs for prediction are the class, position, and velocity of object of interest which can be obtained through 3D object detection approaches. However, often times, intermittent noise and/or loss in detections is observed pointing to the need for a robust estimation scheme. Traditional one-step lookback-based filtering and estimation approaches do not perform well due to a lack of sufficient prior information and simplistic model assumptions. On the other hand, most data-driven approaches do not offer any explicit embeddings of physical motion models or constraints leading to lack of generalizability in unseen scenarios.

To this end, a constrained moving horizon state estimation (MHE) approach to estimate an object's states with respect to a global stationary frame including position, velocity, and acceleration that are robust to intermittently noisy or absent sensor measurements is proposed. Utilizing a computationally light-weight fusion of a Convolutional Neural Network (CNN)-based 2D object detection algorithm and projected LIDAR depth measurements, the approach produces the required measurements relative to the vehicle frame and combines them with the rotation and translation information obtained via a global positioning and inertial measurement system. The performance of the proposed approach is experimentally verified on an in-house dataset featuring urban crossings, with and without autonomous vehicle motion.

Taking the position and velocity estimates as inputs, three key observations in microscopic agent-agent behaviour are incorporated for motion prediction namely – inclination to maintain direction of heading for pedestrians and follow lane centers for vehicles when

free of surrounding agents, tendency to maintain heading and speed unless a collision is anticipated, and most importantly social interaction demonstrating collision avoidance. Traditionally, a fixed model or a model chosen from a fixed set of models is used for modelling future behaviour. These models are applicable to a variety of scenarios, however, they have an inherent bias and may lead to inaccurate predictions. On the other hand, purely data driven approaches suffer from a lack of holistic set of rules governing predictions and hence do not generalize well to a variety of scenarios.

To address these issues, a novel potential field-based model predictive control (MPC) algorithm, MPC-PF, is proposed incorporating social interaction in a single cost function. Simulation results on a variety of scenarios including pedestrians and vehicles approaching directly head-on or otherwise show accurate predictions for a long future horizon. Furthermore, detailed qualitative and quantitative evaluation on a large public motion prediction dataset demonstrates state-of-the-art performance achieved by the proposed approach. Lastly, the potential field-based notion is integrated in a hybrid data driven Deep Deterministic Policy Gradient (DDPG) reinforcement learning (RL) agent, termed RL-PF, with a reward function governed by the potential field and is a valuable direction for further research and experimental validation.

Acknowledgements

Research is rarely the work of a single person and this thesis is no exception; I would like to thank everyone who made this thesis possible. First and foremost, I would like to thank my research supervisors, Prof. Amir Khajepour and Prof. Ehsan Hashemi, for their unwavering support, encouraging guidance, and assuring confidence over the past four years. They broadened and shaped my thoughts not only for research but on the practical side of tasks and life in general which I value the most. In addition, I want to thank them for giving me the platform and learning experience to lead the team in all of the research projects I was involved in at both the Mechatronic Vehicle Systems Lab and NODE Lab. Leading the WATonoBus project and taking it to daily operation via the ministry's pilot program from the grass roots level has been a great experience and needs a separate thesis of its own. In addition, leading several research and development projects from General Motors along with the opportunity to intern at GM in the summer, the outdoor and indoor sensor node projects, and several research projects at NODE Lab with their autonomous vehicle have been equally enriching.

I would also like to thank Prof. Yang Shi, Prof. Soo Jeon, Prof. Arash Arami, and Prof. Stephen L. Smith for serving on my doctoral committee. They provided critical feedback and useful ideas for my research. This helped me view my work from different perspectives.

I am deeply grateful to my friends at MVS Lab. Specifically, thanks to Aaron Sherratt who I spent the most amount of time with during my PhD and learned that work will keep continuing to pile up, but if you do not have fun along the way, it only becomes more difficult. Thanks to Ruihe (Bolt) Zhang who has been with me in my journey since nine years right from undergraduate studies at University of Toronto and whom I have learned a lot from for academics and life in general. Thanks to Joseph (Chen) Sun, who although I have known since only the last 2 years, I have learned a lot from and has been a great friend. Thanks to my dear friends Fernando Barrios, Ehsan Mohammadbagher, Amin H Korayem, Assylbek Dakibay, Yubiao (Gary) Zhang, Yukun Lu, Shucheng (Bruce) Huang, Lucas Botelho, and Mehdi Abroshan for starting the journey with me; I have learned a lot from you in sharing a passion for learning new ideas quickly. Thanks also to Pouya Panahandeh and Ahmad R Alghooneh; we have been great friends and have learned a lot from each other and always kept a curious mind alive. Lastly, thanks to Ben MacCallum, Ted Ecclestone, Minghao Ning, Yaodong Cui, Jeff Graansma, Adrian Neill, and Michael Duthie we have all worked well together as a team. Finally, thank you to the many many people I have not named but who have nonetheless helped me along my path.

Finally, I would like to immensely thank my parents, Archana and Pratik, and my brother, Om, whose immense hard work, selfless dedication, and utmost sacrifice, gave me the opportunity to pursue the journey of life. You have been my life coaches, guiding lights, therapists, and greatest comedians throughout everything. I owe everything I have achieved to you and I hope to share whatever I have learned with others.

I gratefully acknowledge the support from the Ontario Graduate Scholarship and QEII-GSST in funding the work detailed in this thesis.

Dedication

To my parents and my brother.

To everyone else; the world is one family.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

An overwhelming amount of today's traffic accidents are caused by the most failure-prone component of the vehicle: the driver [4]. Despite the rise of recent driver assistance features in consumer cars, automakers have shifted their focus from injury prevention to accident avoidance altogether, thanks to the notion of self-driving vehicles (SDV) [5]. One of the many limiting factors preventing the full scale implementation of these autonomous vehicles on roads is the lack of reliable techniques to perceive surrounding objects and subsequently predict their intent to make safe decisions in complex driving settings such as stop-sign controlled intersections requiring co-ordination between vehicles [6, 7, 8]. Thus, trajectory prediction for safe autonomous decision making is a primary subject of research in the autonomous driving field. The objective of the approach presented in this thesis is to provide reliable estimation of the position and velocity of objects surrounding the SDV and thereafter predicting their future trajectory to enable safe decision making. In this chapter, the primary motivation behind state estimation and motion prediction is presented in Section 1.1, followed by definition of scope and objectives for the proposed approach in Section 1.2. Furthermore, the primary contributions are outlined in Section 1.3 followed by an overview of the modules constituting the proposed approach in Section 1.4. Lastly, an outline of the core chapters is presented in Section 1.5.

1.1 Motivation

One of the sub-modules required for human level decision making is behaviour prediction of surrounding objects such as pedestrians and vehicles. Anticipation of the intent of relevant

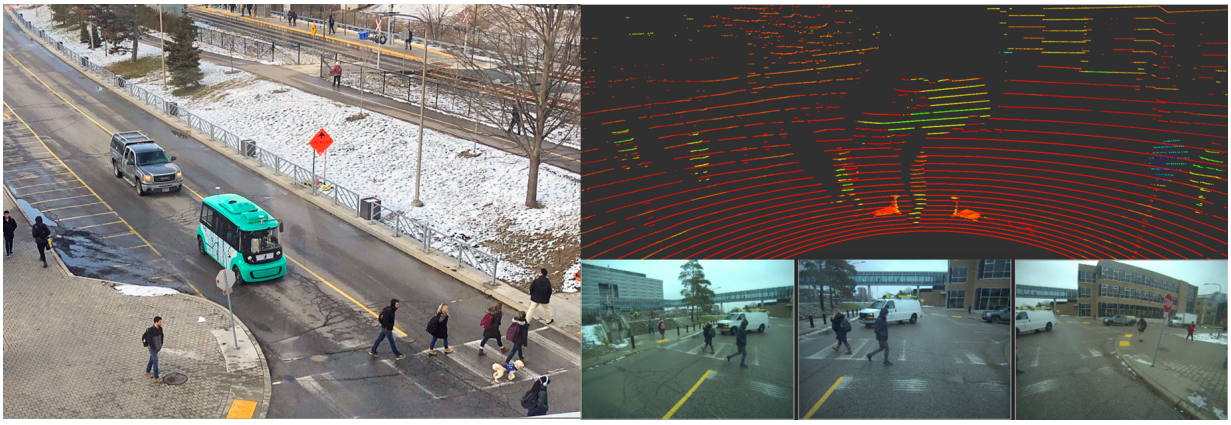


Figure 1.1: Illustration of a typical scenario at an urban intersection. Estimating the position, velocity, and overall intent of objects surrounding an autonomous vehicle is critical for decision making.

objects around the SDV, for instance in the scenario shown in Figure 1.1, is critical to not only ensure safe operation of the vehicle but also enable smooth human like maneuvers [9].

To predict the trajectory of objects and gather situational awareness, estimation of the position and velocity plays an essential role, especially in dense urban areas. However, there are still many technical challenges in estimating a object's position and velocity states. Specifically, model and environmental uncertainties as well as measurement noise in detection modules hinder development of reliable object localization algorithms [10, 11].

Moreover, once the current states of an object are estimated, the primary objective in object behaviour prediction is to estimate the future states, that is position and velocity with respect to the vehicle, in a finite future time period. The main challenge in predicting the behaviour of objects is determining an appropriate model that considers several agent-agent and agent-space interactions noting that the underlying model is generally dynamic and unknown. There are many variables that influence the future trajectory of an object such as the number of possible trajectories at an intersection, the road and lane boundaries near the object, and the possible presence of other objects. Given that object behaviour is often influenced by the interaction of the object with surrounding objects, the space around them, and situational cues, a practical approach should present a structured approach for modelling these intuitions on trajectory prediction while being capable of running efficiently.

1.2 Scope and Objectives

The goal of this thesis project is to ultimately develop a socially-aware object motion prediction algorithm that first performs object detection and generates accurate position and velocity estimates; it subsequently utilizes these as inputs to capture social interactions amongst objects and their surroundings in dense urban settings for safe autonomous driving systems. The approach must remain applicable to all types of objects such as pedestrians and vehicles with modifications only required for their nominal velocity, potential field shape, and expected drivable regions.

Accordingly, the two primary tasks for this work are: (I) estimation of 3D position and velocity of objects and (II) predicting the future motion of objects for a finite time period to determine intent and time to collision.

I. Estimation of 3D position and velocity of objects

The primary objective of this task is to solve three particular challenges in object state estimation namely: (a) intermittent measurement noises and losses, (b) dependence of the object motion, as observed from the SDV, on the motion of the SDV, and (c) situational and physical constraints on object motion states and need for information from other objects related to these constraints.

II. Prediction of future trajectory of objects for a finite time period

There are two main challenges with trajectory prediction of objects namely: (i) anticipating dependence of past trajectories of objects on their future trajectory [12, 13, 14] and (ii) incorporating the effect of social interaction between objects in future trajectory estimates [15].

1.3 Contributions

To address the aforementioned challenges in (a),(b), and (c), a model-based constrained Moving Horizon Estimation (MHE) approach to estimate the object's absolute states is proposed [3]. Our primary contribution is the design and implementation of the MHE approach enabling the following:

- Handling intermittent sensor **measurement losses** and enabling smooth 3D object motion estimation.
- **Compensating** effect of SDV's motion on observed relative motion of objects.

- **Co-estimation** of motion parameters (velocity and acceleration in 3D) along with position tracking through a **single** convex cost function.
- Imposing **constraints** over states and motion parameters yielding practically sound 3D estimates.

To address challenges (i) and (ii), a potential field (PF)-based trajectory predictor: MPC-PF is proposed [2, 1]. It is named MPC-PF as the cost function used draws parallels with Model Predictive Control (MPC), although there is no control action involved here. An overview of the approach is shown in Figure 1.2. MPC-PF enables the following:

- **Incorporating temporal position and velocity information** of objects to inform future trajectory predictions via integration of this algorithm with MHE in a cascaded scheme.
- **Design of potential fields** that model both agent-agent and agent-space information by incorporating map information such as road boundaries, lane boundaries, lane centers, and object positions.
- **Capturing social interactions** amongst objects via design of a novel potential field-based cost function enabling generalizable trajectory prediction while accounting for two main observations on object behaviour, especially for pedestrians:
  (a) Tendency to minimize change in the direction of current heading
  (b) Deviation from direction of current heading in case of (I) anticipated collision with other objects – through a potential field-based cost that does not depend on a direct force-based model, but a rather intuitive heading-based update or (II) to abide with scene cues.

In addition, the design of a hybrid extension of MPC-PF, termed RL-PF, is presented. This Deep Deterministic Policy Gradient (DDPG) reinforcement learning (RL) agent embeds the potential field intuition into a reward function enabling the three key points discussed above.

The proposed approach utilizes outputs from the perception module and subsequently yields predicted trajectories for all objects of interest to the decision making module. It enables intelligent decision making given that perceiving the position and orientation of objects without taking their intentions into account can lead to unsafe conditions.

1.4 Overview of Approach

The overall schematic of the modules involved in predicting the future trajectory of objects is presented in Figure 1.2 below:

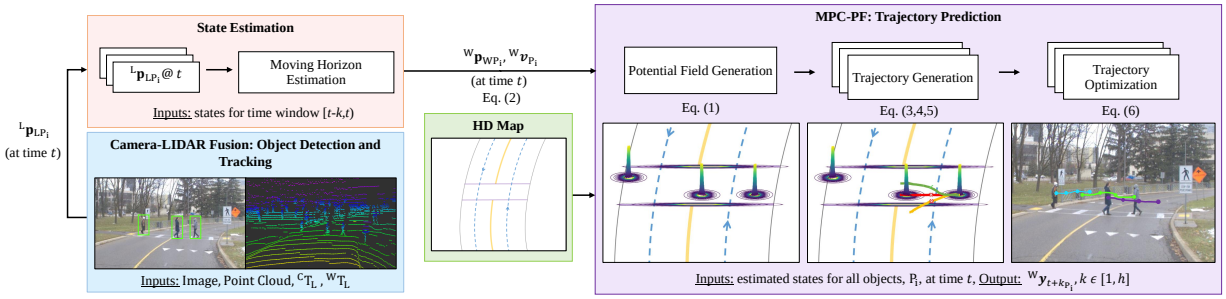


Figure 1.2: **Overview of the MPC-PF model and necessary input modules**: After performing 3D object detection and tracking using camera-LIDAR fusion, smooth and constrained states are obtained via the MHE module. Based on these states and the HD map, a potential field is generated to model surrounding actors and agent-space information. Subsequently, informed by the potential field, trajectories are generated for each agent and the MPC-PF optimization predicts the optimal heading angle at each prediction timestep resulting in a trajectory that incorporates presence of nearby objects and map cues such as crosswalks.

The purpose, inputs, and outputs of each individual module are elaborated in the sections that follow as outlined in the next section.

1.5 Outline

The remainder of this thesis is organized as follows:

In **Chapter** 2, the primary challenges in object state estimation, specifically for perceived states like position and velocity are outlined. Subsequently, a brief discussion on the shortcomings of existing approaches is presented to identify specific research gaps. Similarly, the challenges in object trajectory prediction are identified thereafter. A brief overview on the different types of approaches on motion prediction and a discussion on the state-of-the-art approaches in the literature and their shortcomings are presented subsequently. Finally, the primary research gaps that the proposed approach aims to solve are identified.

In **Chapter** 3, the design details of the constrained MHE approach to estimate an object's states including position, velocity, and acceleration that are robust to intermittently noisy or absent sensor measurements are provided. First, 3D object detection and localization via a computationally light-weight fusion of a 2D object detection algorithm and projected LIDAR depth measurements is elaborately presented. The approach produces the required measurements relative to the vehicle frame and combines them with the rotation and translation information obtained via odometry. Secondly, the MHE system model is defined and the underlying cost function for MHE and the corresponding optimization method is presented.

In **Chapter** 4, MPC-PF is introduced. It is a novel potential field-based trajectory predictor that incorporates social interaction via agent-agent and agent-space considerations and is able to tradeoff between inherent model biases across the prediction horizon. First, the design of agent-agent and agent-space potential fields and visualization of potential fields generated for several scenarios is presented. Subsequently, the system model used for trajectory prediction is defined and the MPC-PF cost function is elaborately presented. Furthermore, the optimization method for the cost function is presented in detail.

In **Chapter** 5, the design of RL-PF, a potential field-based DDPG agent for trajectory prediction, is presented. First, the action and state space for the reinforcement learning agent is defined followed by the design of the reward function considering several factors such as exploration and convergence. Subsequently, the network architecture for the DDPG agent is presented. Furthermore, an evaluation of the agent on two scenarios is presented with results showcasing the applicability of the approach for trajectory prediction.

In **Chapter** 6, the experimental results of the MHE algorithm on a dataset featuring urban crossings, with and without SDV motion, are presented. The results showcase the applicability of the approach in estimating the position and velocity of objects. Furthermore, the results of the MPC-PF algorithm through evaluation on scenarios from the Waymo Open Motion Dataset and a variety of other simulated common urban driving scenarios are presented [16]. The results show that the proposed model is capable of achieving state-of-the-art performance while producing accurate predictions for both short and long term timesteps. In addition, to demonstrate the significance of the model architecture, an ablation study was also conducted and presented.

Finally, in **Chapter** 7, a summary of important findings and contributions as well as proposed topics that prompt future research are presented.

Chapter 2

Literature Review

In this chapter, a review on both model-based and data-driven methods to perform position and velocity estimation as well as trajectory prediction for objects surrounding an SDV is presented. Furthermore, the approach, assumptions, and shortcomings for different methods are discussed. Specifically, in Section 2.1, several position and velocity estimation methods are reviewed and subsequently in Section 2.2 several state-of-the-art trajectory prediction approaches are surveyed. Finally, a summary of the identified research gaps is presented along with challenges that the proposed approach aims to solve.

2.1 Position and Velocity Estimation of Objects

Out of the three particular challenges in object detection and state estimation, challenge (a) is often observed in object detection in general, causing failure in continuous detection of the object of interest across frames, due to deficiency of visual cues, for instance image motion blur, insufficient image resolution, and low lighting conditions, or due to occlusion by other objects. In addition, 3D depth estimation may be contaminated with the depth of other objects in the foreground around the object of interest.

On the other hand, challenge (b) is observed, for instance, when a pedestrian walks straight across the SDV that is gradually coming to a stop. Although from an inertial frame the motion is straight, however the perceived relative motion as observed by the moving SDV is not straight. In such cases, simple model-based estimation schemes that do not incorporate the SDV's motion information are not practical. Thus, recent works have increasingly focused on designing data-driven models that implicitly learn to compensate for SDV's motion [17, 18, 19].

Challenge (c) points to the need for constraints such as the maximum velocity a person can walk or run at, negligible or zero vertical velocity when walking, positive height above the ground, and presence in the vicinity of crosswalk boundaries. Neglecting these constraints lead to non-realistic and unreliable estimates causing model failure.

A summary of various state-of-the-art methods that have attempted to solve some of these challenges including the state estimated by the approach, frame of state observation, model used, and output states is presented in Table 2.1 as an overview.

To address (a), model-based techniques often employ filtering of sensor measurements to reduce noise and utilize a specific system dynamics model to estimate states in absence of measurements. For instance, in [20], a Kalman Filter (KF) approach was implemented and compared to a linear interacting multiple model (IMM) approach, with results indicating that model effectiveness depends on context pointing to the need of using multiple interacting models. Consequently, in [21], the use of IMM along with KF-based on the type of interaction and situational information was presented. The proposed approach was termed IMM-KF. However, switching between different models is yet another task of its own and the estimation performance becomes directly dependant on the performance of the switching model [22]. Another approach based on a Gaussian Process Dynamical Model (GPDM) and a Probabilistic Hierarchical Trajectory Matching (PHTM) scheme was proposed in [23] and compared with the KF and IMM-KF. Utilizing motion information with dense optical flow techniques as opposed to just position information, the approach in [23] exhibits improved performance when the object comes to a sudden stop, however, these non-linear models may need repeated re-tuning making this approach practically infeasible.

Moreover, to tackle (a) and (b), some researchers have proposed using off-board sensor infrastructure such as a Micro-Doppler (MD) radar scheme proposed in [24]. In [25], a set of off-board cameras installed on buildings are utilized for object position tracking and integrated with an on-board navigation module to extend the vehicle's field of view for decision making. However, since use of large-scale infrastructure is impractical, on-board sensor fusion has been widely adopted instead.

Similarly, in recent works, the mapping of observed relative motion to absolute motion is approached via learning trends in data. Most data-driven approaches consider changes in the intent of the objects through a history of motion cues. For instance, in [17], a mixture density layer (MDL)-based supervised long short-term memory (LSTM) network was proposed to produce trajectory estimates based on past actions. Another approach proposed in [26] took this LSTM-based scheme further to extract the SDV and the object states in an attempt to address challenge (b). On the other hand, in [22], a change-point

detection algorithm together with an unsupervised learning scheme for a Gaussian Mixture Model (GMM) was developed for detection of changes in intent for trajectory prediction.

Furthermore, to address (c) a social LSTM and a hub-based LSTM is proposed in [15] and [18] jointly estimating the states of multiple surrounding objects using learned interactions at crosswalks. Similarly in [27], a set of trajectories navigated by surrounding objects is utilized. Moreover, in [28], spatial location information is used to capture multi-modal posteriors over future trajectories using a discrete residual flow convolutional neural network (CNN). Moreover, in [19], a Siamese CNN-based feature extractor is proposed for association of trajectories for position tracking. This approach enables 3D spatial tracklet generation, however, it does not address (c) as it is tested for a stationary SDV.

Table 2.1: A survey of methods used for position and velocity estimation of objects.

Method	Year	Estimated State	Frame	Model Used	Output
IMM-KF [21]	2020	position	local	multiple dynamic models	current position
SiameseCNN [19]	2019	position	local	CNN	position
MDL-LSTM [17]	2018	position	global	particle filter+LSTM	position
MD Radar [24]	2016	heading	local	autoencoder	heading
GMM [22]	2015	position	local	changepoint detection+IMM	position
PHTM [23]	2013	position	local	GPDM+matching	current position+intention
Bayesian-KF [20]	2013	position	local	Bayesian KF	current position+trajectory
SVM [25]	2013	position	global	SVM	position

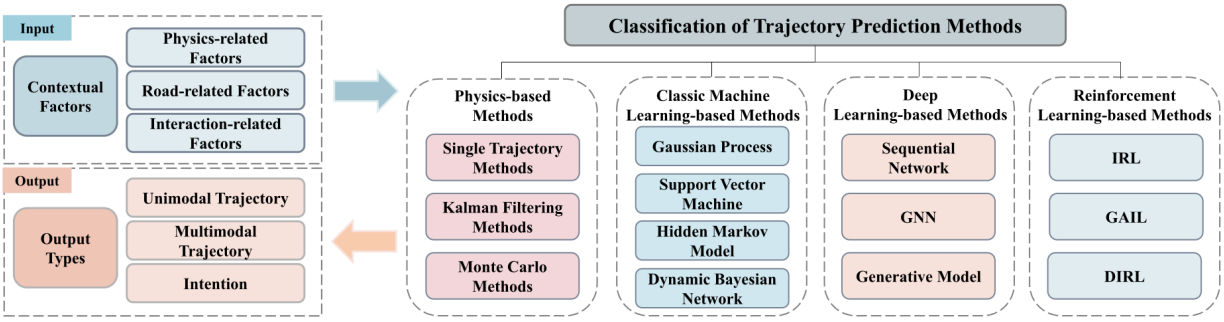


Figure 2.1: Overview of approaches taken in the literature for trajectory prediction.

2.2 Trajectory Prediction of Objects

Trajectory prediction of objects has been studied widely and can be grouped into works that develop an understanding of agent-space interactions and agent-agent interactions. Works focusing solely on agent-space interactions are primarily used for tracking and data association after detection is performed [29, 30, 31, 32]. However, the focus of this thesis is to model both agent-agent and agent-space interactions as it is required to understand the intent of surrounding objects and anticipate any potential collision with the SDV. An overview of different approaches taken in the literature for trajectory prediction is presented in Figure 2.1 as has been recently surveyed in [33]. In addition, a summary of various state-of-the-art methods including the model used for encoding inputs such as the states and scene information and for interaction is presented in Table 2.2.

To tackle (i), traditional model-based approaches have attempted to predict object trajectories using KF and a particle filter wherein the prediction model is usually fixed or chosen from a fixed set of models such as an IMM with a constant velocity, constant acceleration, or constant turn model [34, 35, 36, 20]. Due to the use of a fixed set of models to predict the trajectory of objects, these approaches work well in general, but fail to be accurate in specific scenarios that objects encounter at urban intersections due to inherent biases in the model.

Furthermore, a popular approach to tackle (ii) is the use of the social force concept that models the behavior of objects based on attractive forces, for instance to model groups of pedestrians walking in the same direction, and repulsion forces to avoid collisions with approaching objects. In [37, 38, 39], for instance, a physical repulsion force based on a decaying exponential function is introduced. The amount of force depends on the distance between nearby objects while an attraction force is imposed based on the distance to

the estimated goal location of the object of interest. These approaches capture social interactions, however they solely rely on the distance between any given pair of objects to generate predictions. Thus, anticipation of the repulsion forces once the objects pass each other needs to be handcrafted by reducing weights for the forces near the end of the prediction horizon. In addition, a direct relation between the anticipated position of surrounding objects from their velocity (both speed and heading) and the adaptation of the repulsion and attractive forces that incorporates complex scenarios can not be defined easily due to the collision avoidance being expressed in a social force model.

Several field-based approaches have been proposed to solve different problems in robotics through attractive and repulsive fields or through modeling cost via potential fields [40, 41, 42]. However, most of these approaches are used in navigation and motion planning and none have been designed specifically for object trajectory prediction for autonomous vehicles [43, 44, 45, 46, 47, 48]. Consequently, these field-based approaches are not designed to incorporate map information such as road and lane boundaries, lane centers, crosswalks, or stop-signs. To this end, a potential field and an interaction-aware trajectory predictor that can accomplish this task is designed.

Deep supervised CNNs and Recurrent Neural Networks (RNNs) have thus been developed to implicitly learn the underlying non-linear behaviour model in an end-to-end manner, that is taking the image frames as an input and predicting potential trajectories while capturing social interaction. For instance, [15, 18, 49, 50, 51, 52] propose a LSTM-based socially aware recurrent networks to perform prediction. Such approaches primarily use polylines or raster images to encode scene information and states [53, 54, 55, 56, 57, 58, 59]. In addition, recent trends indicate an increase in use of Graph Neural Networks (GNN) and transformer networks as encoder or decoder networks as opposed to simple Multi Layer Perceptions (MLP) [60, 61]. Moreover, attention networks have become popular to capture interaction between agents [60, 61, 57]. However, the performance of these networks depends on the availability and size of ground truth labels which is often difficult to obtain in large amounts. In addition, these approaches inherently depend on collection of data that covers all possible object interactions to generalize and consistently generate sound predictions in specific scenarios on unseen datasets. Thus, due to the lack of an explicit representation of social interaction with other agents, these approaches do not generalize well to the various different scenarios that autonomous vehicles encounter and do not scale well with increasing number of objects in the scene.

Given these limitations, reinforcement learning-based prediction schemes have started to become an active area of research. This is in particular due to the ability of reinforcement learning agents to incorporate both data driven actors while utilizing model-based rewards offering a fusion of the aforementioned approaches. Specifically, the absence of large amount

of labelled data, points to the importance of the designing the reward function as it is the primary source for the agent during learning. Since the reward model is generally unknown, this problem can be modelled as: an (i) Inverse Reinforcement Learning (IRL) problem or (ii) the reward model can be assumed to be based on heuristics. To incorporate a measure of uncertainty in the expected reward, various works have proposed variants of the maximum entropy IRL formulation to calculate the reward as a function of the input features estimated by a LSTM to model effect of multi-agent social interactions [62, 63, 64, 65, 66, 67, 68]. In both approaches (i) and (ii), the action selection is based on the reward model that is computed based on the surrounding agents. However, IRL-based reward functions do not generalize well to unseen scenarios due to the direct dependence of resulting reward function on the states observed during training.

Thus, the insights from prior work suggest that model-based approaches can generalize well to large range of scenarios without the need of large, rich datasets but can lead to sizeable biases in specific scenarios. At the same time CNN- and RNN-based approaches can provide predictions with comparatively unbiased estimates for datasets on which they are trained, but may not generalize well to other settings.

Table 2.2: A survey of recent methods in motion prediction.

Method	Year	Context Encoder	State Encoder	Interaction Module	Decoder	Output
MTR-A [60]	2022	polyline	polyline	attention	transformer	state poly
golfer [53]	2022	polyline	polyline	MLP	MLP	states
Multipath++ [54]	2021	polyline	LSTM	RNNs+maxpool	MLP	control poly
DenseTNT [69]	2021	VectorNet	VectorNet	VectorNet	MLP	state poly
ReCoAt [55]	2021	polyline	polyline	attention	LSTM	state poly
SimpleCNNOnRaster [56]	2021	raster	raster	CNN	MLP	state poly
LaneGCN [61]	2020	GNN	1D conv	GNN	MLP	states
VectorNet [57]	2020	polyline	polyline	attention+maxpool	MLP	states
TNT [58]	2020	polyline	polyline	attention+maxpool	MLP	states
Trajectron++ [59]	2020	raster	LSTM	RNNs+attention	GRU	controls
SocialLSTM [15]	2016	None	LSTM	spatial pooling	LSTM	states

2.3 Summary

In this chapter, a review of the methods on position and velocity estimation of objects and on trajectory prediction was presented. A summary of the approaches is presented in Tables 2.1 and 2.2.

Traditional estimation approaches use a fixed or a set of fixed kinematic models to perform estimation and often fail to address intermittent loss in measurements or are unable to decouple SDV motion from object motion due to one-step lookback-based filtering. On the other hand, data-driven approaches lack a physical model about object motion and often suffer from scalability issues in unseen scenarios.

Similarly, trajectory prediction models solely based on constant velocity or social force have an inherent bias and may lead to inaccurate predictions across the prediction horizon whereas purely data driven approaches suffer from a lack of a holistic set of rules governing predictions.

To this end, the proposed approach attempts to bridge this gap by design of a moving horizon-based estimator for position and velocity of objects that can address intermittent measurement loss, decouple SDV motion from object motion, and constrain model output to physically sound and reliable estimates. More importantly, through the design of MPC-PF, the temporal position and velocity estimates are used along with agent-agent and agent-space potential field to capture social interactions amongst object while remaining generalizable to several scenario and yielding state-of-the-art results.

Chapter 3

MHE: Moving Horizon Object Position and Velocity Estimation

In this chapter, the constrained MHE estimator is elaborated upon. In Section 3.1, the global object measurement model is defined in Section 3.1.1 and subsequently fusion of 2D object detections and LIDAR point cloud is outlined in Section 3.1.2 to obtain raw 3D positions of objects. Thereafter, in Section 3.4, first the motion and measurement models are defined in Section 3.4.1 and the core MHE cost function is established with explanation on the significance of each of the three terms in Section 3.4.2. Lastly, derivation of the cost function to the quadratic programming (QP) form is presented for optimization in Section 3.4.3.

3.1 3D Object Detection and Localization

Perceiving surrounding objects is a precursor to performing any trajectory prediction. Multiple perception approaches exist in the literature utilizing 2D or 3D bounding box detection for classification. With only monocular vision, accurate depth estimation is often challenging. To this end, various researchers use depth information from LIDAR directly to cluster points and obtain a 3D bounding box position estimate of objects with respect to the SDV. This approach inevitably requires search through the entire point cloud. Given LIDAR point cloud processing is computationally expensive and difficult to realize in real-time without cutting edge hardware, the aim of the approach proposed in this thesis proposal is to provide a computationally light solution for real-time implementation. Additional details on the enhancing runtime efficiency is presented in Appendix A.

Specifically, the approach presented here utilizes 2D detections in the image and the corresponding bounding boxes to narrow down the depth clustering search and fuse depth information from the LIDAR. This is different from the approach aforementioned in that the search for depth clustering is informed by the bounding boxes and hence reduces computational cost.

3.1.1 Global Object State Measurement Model

To locate objects in the image frame, a Deep Neural Network (DNN)-based object detector is used. The DNN performs two primary tasks: (1) 2D bounding box regression and (2) Classification of the object within. The output of the DNN is the position of the center of the bounding box with respect to the top left corner of the image frame.

The subsequent task is to find the objects's 3D position relative to the camera frame ${}^{C}\mathbf{p}_{CP}$. Figures 3.1 and 3.2 illustrate the coordinate frames that are considered to achieve this task. The depth information is extracted by searching within the bounds of the bounding box thereby reducing computational cost. This is further illustrated in Section 3.1.2. The ultimate aim of 3D localization of the object with respect to the world frame can be achieved by using the central projection camera model as well as through odometry information via following equations:

$$z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}^{I} = K \, {}^{C}\mathbf{p}_{CP} \;, \; K = \begin{bmatrix} f_u & s & P_u & 0 \\ 0 & f_v & P_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.1}$$

where ${}^{C}\mathbf{p}_{CP}$ can be obtained as shown below:

$$\left[\begin{array}{c} {}^{C}\mathbf{p}_{CP} \\ \hline 1 \end{array}\right] = \left[\begin{array}{c|c} {}^{C}R_{W} & {}^{C}\mathbf{p}_{CW} \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c} {}^{W}\mathbf{p}_{WP} \\ \hline 1 \end{array}\right] \tag{3.2}$$

Here (u, v) forms the position (in pixels) of the center of the bounding box with respect to the the image frame and z is the depth (in meters) of the corresponding object with respect to the camera frame. K is the camera intrinsic matrix and ${}^{C}R_{W}$ and ${}^{C}\mathbf{p}_{CW}$ are the rotation matrix and the translation vector denoting the rotation and position vector of the world frame with respect to the camera frame respectively. f_u and f_v are lens focal lengths in pixels, P_u and P_v are x and y coordinates of the optical center in the image frame, and s is the skew coefficient. According to Figure 3.2, the object's position is given by:

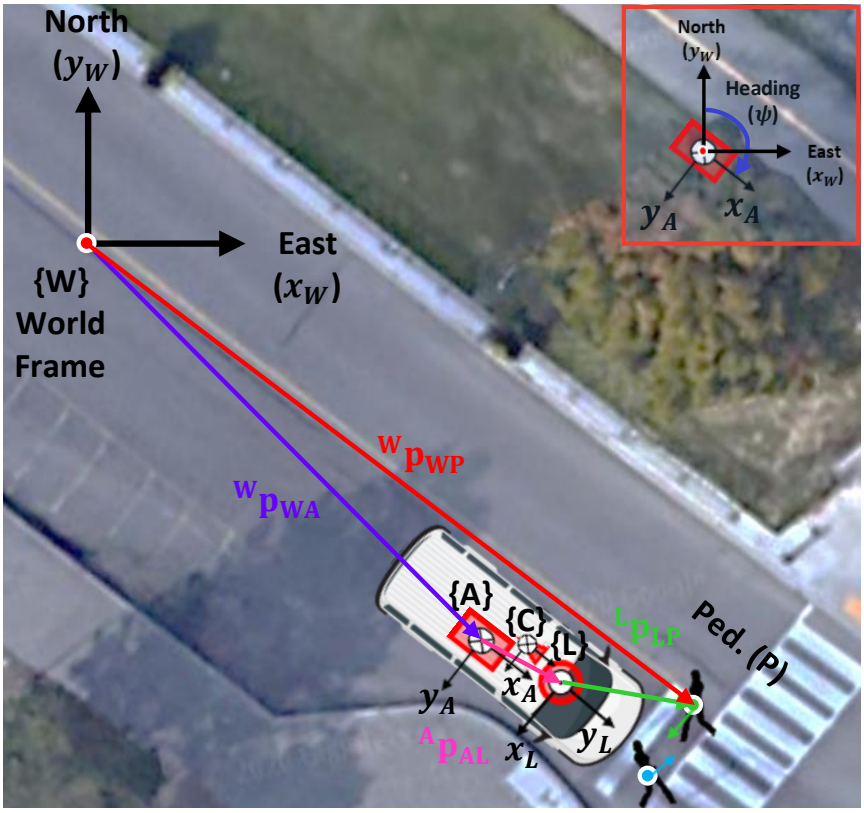


Figure 3.1: Generic scenario depicting two pedestrians in front of the SDV and sensor co-ordinate frames involved. Note that here, there are four frames shown: the world frame **{W}**, the GPS/IMU frame **{A}**, the LIDAR frame **{L}**, and the camera frame **{C}** attached to SDV.

$$ {}^{W}\mathbf{p}_{WP} = {}^{W}\mathbf{p}_{WC} + ({}^{W}R_{C})\ {}^{C}\mathbf{p}_{CP} \tag{3.3} $$

where ${}^{W}\mathbf{p}_{WP}$ and ${}^{C}\mathbf{p}_{CP}$ are the object's position in the world frame and camera frame respectively, ${}^{W}\mathbf{p}_{WC}$ is the camera's position in the world frame and ${}^{W}R_{C}$ is the rotation matrix from the camera frame to the world frame.

Combining Equations (3.1) and (3.3), the following is obtained:

$$ {}^{W}\mathbf{p}_{WP} = z({}^{W}R_{C})K^{-1}\ {}^{I}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} + {}^{W}\mathbf{p}_{WC} \tag{3.4} $$

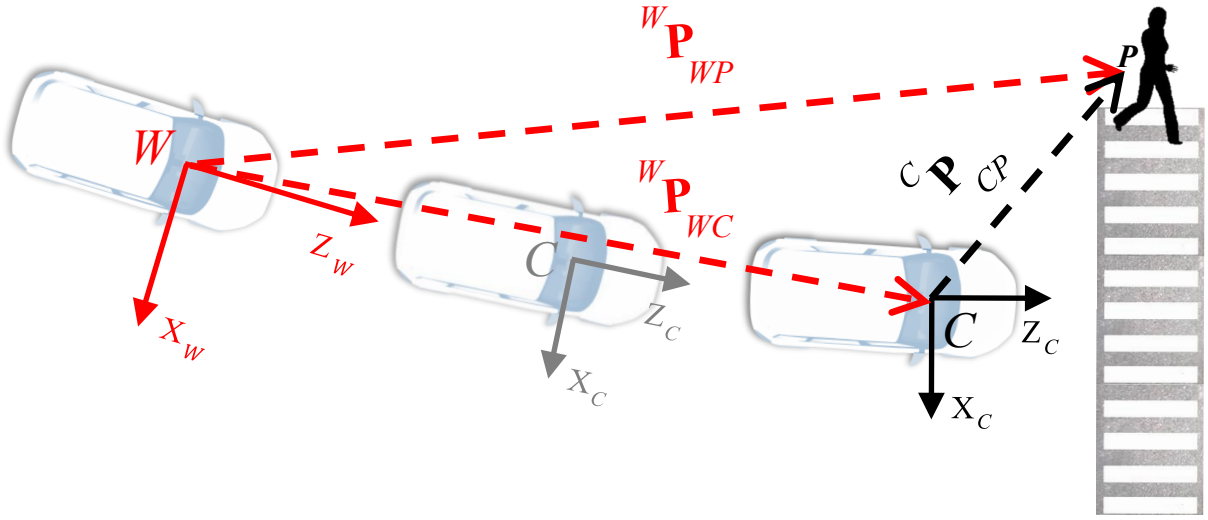


Figure 3.2: Overview of coordinate frames. When the SDV starts for the first time, the corresponding IMU frame is considered as the world frame $[X_W, Y_W, Z_W]$. Afterwards, the position of the object, in this case it is a pedestrian, ${}^{C}\mathbf{p}_{CP}$ is measured in the camera frame $[X_c, Y_c, Z_c]$. ${}^{W}\mathbf{p}_{WC}$ denotes the position of the camera expressed in **{W}**.

In Equation (3.3), ${}^{W}R_{C}$ and ${}^{W}\mathbf{p}_{WC}$ are deduced from the SDV's odometry information via on-board GPS and IMU as well as the rotation matrix ${}^{W}R_{A}$ and the extrinsic translation vector ${}^{A}\mathbf{p}_{WA}$ as shown below:

$$ {}^{C}T_{W} = \begin{bmatrix} {}^{C}R_{W} & {}^{C}\mathbf{p}_{CW} \\ 0 & 1 \end{bmatrix} = {}^{W}T_{A}{}^{A}T_{C} = \begin{bmatrix} {}^{W}R_{A} & {}^{W}\mathbf{p}_{WA} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{A}R_{C} & {}^{A}\mathbf{p}_{AC} \\ 0 & 1 \end{bmatrix} \tag{3.5} $$

In doing so, the position vector, ${}^{W}\mathbf{p}_{WA}$, is obtained through GPS and the rotation matrix, ${}^{W}R_{A}$, via the heading, ψ, from IMU. In addition, knowing the extrinsic transforms between the sensors allows expressing the global position of any object in any of the listed sensor frames.

3.1.2 Efficient Fusion of 2D Object Detections and Depth

Efficient fusion of object depth and camera data involves projecting all 3D points obtained from a depth sensing unit such as a LIDAR, RADAR, or stereo camera into the image frame and constraining the depth search to within the horizontal and vertical bounds of

Figure 3.3: Illustration of several frames used for checkerboard camera intrinsic calibration.

the predicted bounding box for the object in consideration. In this work, a LIDAR is used to obtain this information. The extrinsic transformation matrix from the LIDAR frame, **{L}**, to the camera frame can be obtained via multi-point correspondence between LIDAR and camera points utilizing the following relation:

$$ {}^{I}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z}\, K \left[\begin{array}{c|c} {}^{C}R_{L} & {}^{C}\mathbf{p}_{CL} \\ \hline 0 & 1 \end{array}\right] {}^{L}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.6} $$

where ${}^{C}R_{L}$ and ${}^{C}\mathbf{p}_{CL}$ denotes the rotation from the LIDAR frame to the camera frame and the translation from the camera frame to the LIDAR frame, respectively. Subsequently, a histogram of the depth values of points within the bounding box are generated and the average of the nearest 20% of the depth values is assigned as the depth (determined and tested empirically), z, of the object with respect to the camera frame. Thus, ${}^{L}[x, y, z, 1]^{T}$ obtained position of the object with respect to the LIDAR frame (${}^{L}\mathbf{p}_{LP}$) in homogeneous form. Thus the global position of the object can be obtained as follows:

$$ {}^{W}\mathbf{p}_{WP} = {}^{W}\mathbf{p}_{WA} + {}^{A}\mathbf{p}_{AL} + {}^{L}\mathbf{p}_{LP} \tag{3.7} $$

Figure 3.4: Illustration of several frames showing the projected LIDAR point cloud on the image captured. Despite the object moving to different extents of the image, the projection remains accurate.

Thus, this approach reduces computational cost of depth association search by limiting the search space via the detected 2D bounding box and allows for real-time localization of the object with respect to the world frame.

3.2 Camera Intrinsic Calibration

The intrinsic parameters of a camera, required in Equation (3.1), can be obtained via checkerboard camera calibration. The corners of the checkerboard pattern are easily detectable control points that can be used to map pixel space to measurements in a world coordinate frame given that the prior size of the squares in the checkerboard pattern is known. The checkerboard pattern also allows for estimation of distortion parameters due to barrel distortion.

The intrinsic calibration yields the estimated f_u, f_v, s, P_u, and P_v parameters required in Equation (3.1). An illustration of the checkerboard calibration performed to obtain the intrinsic parameters is shown in Figure 3.3.

3.3 Camera-LIDAR Extrinsic Calibration

Extrinsic calibration for the camera and LIDAR is required to obtain ${}^{C}R_{L}$ and ${}^{C}\mathbf{p}_{_{CL}}$ required in Equation (3.6). Given that there are six unknown degrees of freedom required to obtain these two quantities, the LIDAR points can be projected onto the camera via a preliminary guess and can be improved either manually or algorithmically via the projection error that can be both qualitatively identified via the projection image or quantitatively via checkerboard corner position verification. This approach does not need specialized calibration boards or shapes to be fabricated.

Such an extrinsic calibration was performed and an illustration of the process is depicted in Figure 3.4. A **tool** developed by the author for achieving this is available at https://github.com/Neel1302/lidar-camera-calibration.

3.4 Moving Horizon Estimation

Aforementioned measurement uncertainties point to the need of an estimation scheme that handles (i) Intermittent losses of measurements, particularly 2D detections and (ii) Intermittent erroneous measurements, particularly those from LIDAR, e.g., due to signs that are closer to the SDV than the object in the bounding box. To this end, a constrained moving horizon estimation scheme to address (i) and (ii) is proposed.

3.4.1 System Model

Since it is common to encounter motion blur in images, it is expected that the object cannot be detected in all consecutive frames. In such a situation where the input images are noisy, it is necessary to estimate the position and the velocity of the object in order to make predictions about the path. However, this requires a dynamic model of the object's motion and the exact model of the object's motion is generally unknown.

It is assumed that the object follows a constant acceleration model. This assumption is valid for a short timestep. With the aim of estimating the positon and the velocity of the object with respect to a stationary frame in 3D, the state vector is formed as $\mathbf{x} = [x \quad y \quad z \quad u \quad v \quad w]^{\top}$ consisting of the position and velocity of the object with respect to the world frame. The object acceleration vector, $\mathbf{a} = [a_x \quad a_y \quad a_z]^{\top}$, is also to be estimated and is considered in the following dynamical model of the system (discretized via zero-order hold: constant acceleration assumption):

$$\begin{aligned} \mathbf{x}_{k+1} &= \overbrace{A\mathbf{x}_k + D\mathbf{a}}^{s_k} + \mathbf{w}_k \\ \mathbf{y}_k &= \underbrace{C_k \mathbf{x}_k}_{h_k} + \mathbf{v}_k \end{aligned} \tag{3.8}$$

where,

$$\mathbf{y}_k = [u\, v\, 1]^\top - K\, {}^C\mathbf{p}_{CW}, \quad C_k = K\,({}^C R_W)\, C \tag{3.9}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \; D = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}^\top$$

In addition, $\mathbf{w}_k$ and $\mathbf{v}_k$ represent the process noise and measurement noise, respectively. Note that there is no control input for the system since the intentions and decisions of the object considered are not observable. The optimization problem is formulated in the next section.

3.4.2 Design of MHE Cost Function

Given the system model in Equation (3.8), the objective is to minimize the following convex cost function with the minimization variables $\mathbf{x}_{k-L+1}, \mathbf{x}_{k-L+2}, ..., \mathbf{x}_k$ and $\mathbf{a}$:

$$\min_{\mathbf{x}_j, \mathbf{a}} \left[\overbrace{\left\| \begin{matrix} \mathbf{x}_{k-L+1} - \bar{\mathbf{x}}_{k-L+1} \\ \mathbf{a} - \bar{\mathbf{a}} \end{matrix} \right\|^2_{P_L}}^{\text{Arrival Cost}} + \overbrace{\sum_{j=k-L+1}^{k} \|y_j - \hat{y}_j\|^2_{V_j}}^{\text{Mesurement Residual}} + \overbrace{\sum_{j=k-L+1}^{k-1} \left\|\mathbf{x}_{j+1} - s_j\right\|^2_W}^{\text{Motion Model Residual}} \right]$$

$$
\begin{aligned}
s.t. \quad & \mathbf{x}_{j+1} = A\mathbf{x}_j + D\mathbf{a} + \mathbf{w}_j, \quad j = k-L+1, ..., k-1 \\
& \mathbf{x}_j^- \leq \mathbf{x}_j \leq \mathbf{x}_j^+, \quad j = k-L+1, ..., k \\
& \mathbf{a}^- \leq \mathbf{a} \leq \mathbf{a}^+, \quad \mathbf{x_j} = [x_j\ y_j\ z_j\ u_j\ v_j\ w_j]^T, \quad \mathbf{a} = [a_x\ a_y\ a_z]^T
\end{aligned} \tag{3.10}
$$

where subscript j denotes the parent variable at time t_j in the finite time horizon window $t\,\epsilon\,[t_{k-L+1}, t_k]$ with horizon length of L. $P_L \in \mathbb{R}^{9\times9}$, $V_j \in \mathbb{R}^{3\times3}$ and $W \in \mathbb{R}^{6\times6}$ are positive semi-definite matrices (note that $\left\|\mathbf{x}\right\|_P^2$ denotes $x^\top Px$ for given positive semi-definite matrix P).

The first term in the cost function, often called *arrival cost*, contains information about past measurements and is a key for the MHE stability [70]; at time t_k, the previously computed solution for timestep t_{k-1} is available, which is $\bar{\mathbf{x}}_{k-L}, \bar{\mathbf{x}}_{k-L+1}, ..., \bar{\mathbf{x}}_{k-1}$ and $\bar{\mathbf{a}}$. The difference between $\mathbf{x}_{k-L+1}$ and $\bar{\mathbf{x}}_{k-L+1}$ as well as $\mathbf{a}$ and $\bar{\mathbf{a}}$ is being minimized.

The second term allows minimizing the error between y_j, the measurements from the sensors, and $\hat{y}_j$, the measurement model output defined in Equation (3.8). The third term in the cost function allows minimizing the difference between the states throughout the horizon and the corresponding motion model prediction term s_j, as defined in Equation (3.8), throughout the horizon.

Lastly, the inequality constraints introduced in Equation (3.10) denote that each of the elements of the vector $\mathbf{x_j}$ and $\mathbf{a}$ must lie inside the defined boundaries.

3.4.3 Optimization: Derivation of MHE Cost Function Into QP

To solve the minimization problem in Equation (3.10), the cost function can be converted to the QP form that can be later solved in real-time using a QP solver algorithm. Accordingly, the problem in Equation (3.10) must be converted into the form expressed in Equation (3.11):

$$
\min_{r_k} J(r_k) = \min_{r_k} \left(\frac{1}{2} r_k^\top H r_k + f^\top r_k \right) \quad s.t. \quad Gr_k \leq W \tag{3.11}
$$

where $H \in \mathbb{R}^{(6L+3)\times(6L+3)}$ is a positive definite matrix and $f \in \mathbb{R}^{(6L+3)}$. Also, $G \in \mathbb{R}^{(6L+3)\times(6L+3)}$ and $W \in \mathbb{R}^{(6L+3)}$ are matrices defining the constraints over states and parameters for the horizon. The new minimization argument, $r_k \in \mathbb{R}^{(6L+3)}$ is the augmented vector of states and the parameters for the finite horizon, defined as:

$$r_k = \begin{bmatrix} \mathbf{x}_{k-L-1}^\top & \mathbf{x}_{k-L+2}^\top \dots \mathbf{x}_k^\top & \mathbf{a}^\top \end{bmatrix}^\top \tag{3.12}$$

Thus, the state and the parameter vector can be expressed in terms of the new augmented state vector as follows:

$$\begin{aligned} \mathbf{x}_j &= R_j^\top r_k \,, \quad R_j = \begin{bmatrix} \mathbf{0}_{6\times 6(j-1)} & \mathbf{I}_6 & \mathbf{0}_{6\times 6(L-j)+3} \end{bmatrix}^\top \\ \mathbf{a} &= R_{\mathbf{a}} r_k \,, \quad R_{\mathbf{a}} = \begin{bmatrix} \mathbf{0}_{3\times 6L} & \mathbf{I}_3 \end{bmatrix}^\top \quad {}_{j=k-L+1,\dots,k} \end{aligned} \tag{3.13}$$

where $\mathbf{0}$ and $\mathbf{I}$ denote *zeros* and *identity* matrices, respectively. Note that each of the three terms in Equation (3.10) is quadratic. Hence, these terms are converted individually into QP form and linearity is used to arrive at the QP form for the entire minimization problem. In doing so, the cost function in Equation (3.10) reduces to Equation (3.14) below:

$$J(r_k) = \frac{1}{2} r_k^\top \underbrace{\left(\bar{H} + H_y + H_s\right)}_{H} r_k + \underbrace{\left(\bar{f} + f_y + f_s\right)}_{f^\top} r_k \tag{3.14}$$

where $\bar{H}$, H_y, and H_s along with $\bar{f}$, f_y, and f_s are the matrices that form H and $f^\top$ respectively and are decomposed to correspond to the arrival cost, measurement cost, and system model cost respectively, which are as follows:

$$\begin{aligned} \bar{H} =& \frac{1}{2} R_{La}^\top P_L R_{La} \\ \bar{f} =& - \begin{bmatrix} \bar{\mathbf{x}}_L^\top & \bar{\mathbf{a}}^\top \end{bmatrix} \left(P_L + P_L^\top\right) R_{La} \\ H_y =& \sum_{j=k-L+1}^{k-1} R_j^\top e_j R_j \,, \quad f_y = \sum_{j=k-L+1}^{k-1} q_j R_j \,, \text{ and} \\ H_s =& R_{j+1}^\top W \left(R_{j+1} - A R_j - D R_{\mathbf{a}}\right) \\ & - \left(R_{\mathbf{a}}^\top D^\top + R_j^\top A^\top\right) \left(W R_{j+1} + A R_j + D R_{\mathbf{a}}\right) \\ f_s =& 0 \end{aligned} \tag{3.15}$$

where, $R_{La} = [R_{k+L-1}^{\top}, R_a^{\top}]^{\top}$, $q_j = -y_j^{\top}\left(V_j + V_j^{\top}\right)C_j$, and $e_j = C_j^{\top}V_jC_j$. Note that P_L is partitioned as:

$$P_L = \begin{bmatrix} P_L^{11} & P_L^{12} \\ P_L^{21} & P_L^{22} \end{bmatrix}, \quad \begin{matrix} P_L^{11} \in \mathbb{R}^{6\times 6} & P_L^{21} \in \mathbb{R}^{3\times 6} \\ P_L^{12} \in \mathbb{R}^{6\times 3} & P_L^{22} \in \mathbb{R}^{3\times 3} \end{matrix} \tag{3.16}$$

Furthermore, to impose constraints on the state parameters and the parameter estimates based on practical expectation for these states, the inequalities are stacked together in $Gr_k \leq W$ as follows:

$$G = \left[G_{\mathbf{x}}^{\top}, G_{\mathbf{a}}^{\top}\right]^{\top} \quad , \quad W = \left[W_{\mathbf{x}}^{\top} W_{\mathbf{a}}^{\top}\right]^{\top} \tag{3.17}$$

where,

$$G_{\mathbf{x}} = \left[R_{k-L+1}^{\top} \ \ldots \ R_k^{\top} \ -R_{k-L+1}^{\top} \ \ldots \ -R_k^{\top}\right]^{\top}, G_{\mathbf{a}} = \begin{bmatrix} R_{\mathbf{a}} \\ -R_{\mathbf{a}} \end{bmatrix} \tag{3.18}$$

$$W_{\mathbf{x}} = \left[\mathbf{x}_{k-L+1}^{+} \ \ldots \ \mathbf{x}_k^{+} \ -\mathbf{x}_{k-L+1}^{-} \ \ldots \ -\mathbf{x}_k^{-}\right]^{\top}, W_{\mathbf{a}} = \begin{bmatrix} \mathbf{a}^{+} \\ -\mathbf{a}^{-} \end{bmatrix} \tag{3.19}$$

Now that the constrained minimization problem is in QP form, the numerical optimal solution for the states and parameters can be computed using a QP solver.

The estimated states are now available in the camera frame $\{\mathbf{C}\}$. A summary of the algorithm is presented in Algorithm 1 and 2 for reference.

Algorithm 1: Main Framework

$L \leftarrow 1$
$r_k \leftarrow zeros(6L+3, 1)$
$k_0 \leftarrow$ first time object is detected
while $k > k_0 + L$ **do**
  if $measurment = true$ **then**
    $mFlag \leftarrow 1$ (Used to set $V[j]$ to 0 when measurement is lost)
    Perform 2D DNN-based object detection
    Project LIDAR points to image and filter points in Bbox
    Compute object depth by Bbox depth averaging
  else
    $mFlag \leftarrow 0$
  end
  for $j = k - L + 1$ *to* k **do**
    $(u, v) \leftarrow$ stack Bbox positions wrt. image frame
    $z \leftarrow$ stack depths obtained from algorithm described in II
    $pose \leftarrow$ stack vehicle pose computed from GPS/IMU measurements
  end
  $r_k \leftarrow$MHE$((u, v), z, pose, \bar{\mathbf{x}}_L, \bar{\mathbf{a}}, \mathbf{x}^+, \mathbf{x}^-, \mathbf{a}^+, \mathbf{a}^-, mFlag)$
  $L \leftarrow \min(L + 1, L_{max})$
end

Algorithm 2: MHE Function

function MHE$((u, v), z, pose, \bar{\mathbf{x}}_L, \bar{\mathbf{a}}, \mathbf{x}^+, \mathbf{x}^-, \mathbf{a}^+, \mathbf{a}^-, mFlag)$**:**
  for $j = k - L + 1$ *to* k **do**
    $V(t) \leftarrow V_0 * mFlag(t)$
  end
  compute $\bar{H}$ and $\bar{f}$ from Eq. (3.15)
  compute H_y and f_y from Eq. (3.15)
  compute H_s from Eq. (3.15)
  if $L < L_{max}$ **then**
    $H \leftarrow H_y + H_s$
  else
    $H \leftarrow \bar{H} + H_y + H_s$
  end
  $f \leftarrow \bar{f} + f_y$
  construct G and W from Eq. (3.17)
  $r_k \leftarrow$quadprog(H, f, G, W)
return r_k

3.5 Summary

In this chapter, a model-based finite horizon MHE framework for online estimation of object position, velocity, and acceleration was presented. First, the approach performs fusion of 2D detections with LIDAR point cloud via depth association wherein the search space is reduced by utilizing 2D detections from vision. Utilizing this scheme, the 3D position measurements for each object can be obtained. The approach utilizes a constrained optimization cost function allowing constraints to be imposed based on physical limitations. In addition, the cost function proposed in Equation (3.10) allows minimization of the state arrival cost, differences between the sensor measurements and the model estimates. In addition, it allows mitigation of intermittent loss in sensor measurements accomplishing the objectives aforementioned. Furthermore, the cost function was converted to the QP form such that it can be solved in real-time using a QP solver algorithm. The overall algorithm is summarized in Algorithms 1 and 2.

Chapter 4

MPC-PF: Potential Field-Based Object Trajectory Prediction Scheme

Given that the position and velocity of objects can be estimated from MHE, they can be utilized as inputs to MPC-PF presented in this chapter. In the following sections, the proposed prediction scheme is outlined. The prediction scheme consists of two main enabling modules: (1) a novel potential field (PF) model based on position and velocity estimates of surrounding objects from MHE and (2) A novel MPC-PF cost function that incorporates social interactions through potential fields to predict the future trajectory of the object of interest. In Section 4.1, the potential field model is defined for agent-agent interactions in Section 4.1.1 and subsequently for agent-space interactions in Section 4.1.2. Thereafter, the system model is established in Section 4.2.1 followed by the core MPC-PF cost function with an explanation on the significance of each term in Section 4.2.2. Finally, the optimization method for the cost function is outlined in Section 4.2.3.

4.1 Modelling Interactions Using Potential Fields

4.1.1 Agent-agent Potential Fields

The potential field concept is a key enabler to capture social interaction. It is analogous to a magnetic repulsion field, however, it is not formulated as a force field, but rather as incurred cost. In general, each relevant object such as a vehicle or a pedestrian can be pictured to have their own potential fields centered at their respective positions expressed in the fixed world frame **{W}**.

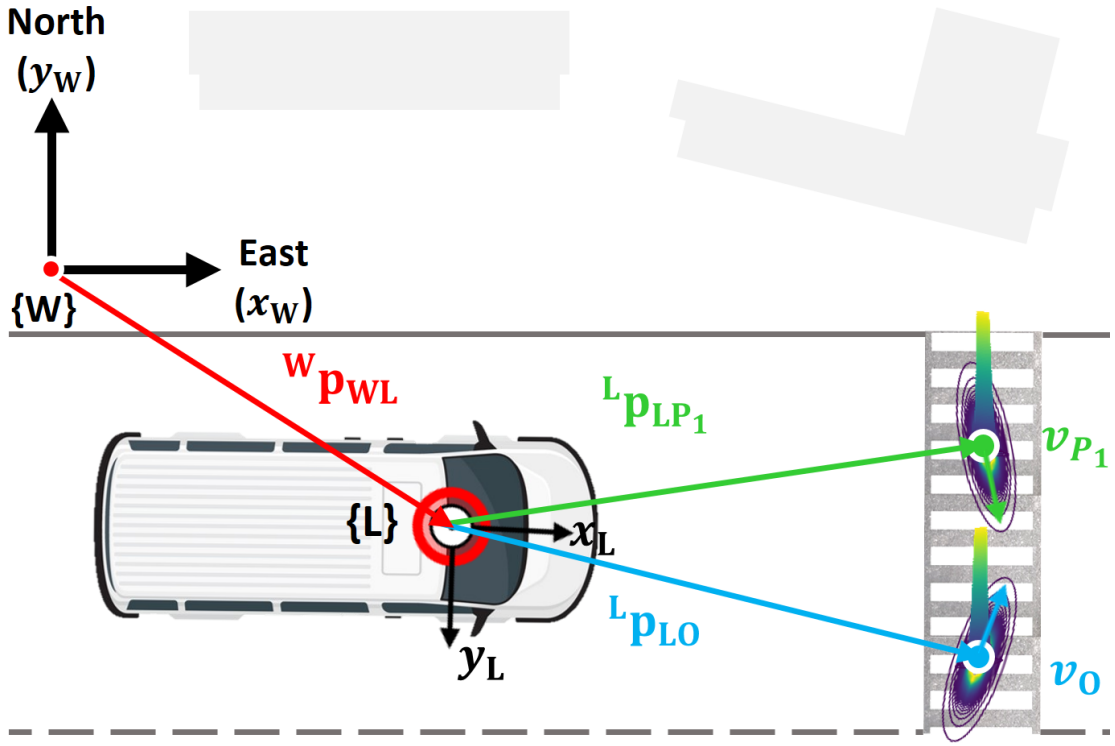


Figure 4.1: A simple scenario depicting two pedestrians at a crosswalk nearby the SDV. The potential fields for both pedestrians are superimposed at their respective positions.

Intuitively, the potential field should be dependent on the **position** at which it is measured, being the most intense (that is repulsive, analogically) at the position of a relevant surrounding pedestrian or vehicle and decreasingly intense as the point of measurement becomes more and more distant from the object. In addition, it must capture the **velocity** (comprising both the speed and direction of heading) of the object of interest as well as the surrounding objects.

Consider the scenario in Figure 4.1. The primary purpose of creating a potential field with the aforementioned properties is to define an effective repulsive zone around any object surrounding the object of interest, in this case say one of the pedestrians shown, and enable prediction of the trajectory so as to navigate around the other objects. The potential field, as shown in Figure 4.1, is the measure of penalty or cost for another object to be present in the vicinity of the object of interest.

To model the intuition offered above, a potential field function which represents the **cost**, U_{t_P}, incurred by an object, O, due to being close to another surrounding object, P,

at a given time, t, is defined as (all vectors expressed in **{W}**):

$$U_{t\,\mathrm{P}}\left(\mathbf{p}_{\mathrm{WP}}, \mathbf{p}_{\mathrm{WO}}, \boldsymbol{v}_{\mathrm{P}}, a, b\right) = \frac{a}{s(X,Y)^{b}} \tag{4.1}$$

where,

$$X = \frac{\left[\ \mathbf{p}_{\mathrm{WP_x}} - \ \mathbf{p}_{\mathrm{WO_x}}\right] \cos\theta_{\mathrm{P}} + \left[\ \mathbf{p}_{\mathrm{WP_y}} - \ \mathbf{p}_{\mathrm{WO_y}}\right] \sin\theta_{\mathrm{P}}}{\boldsymbol{v}_{\mathrm{P_x}}}$$

$$Y = \frac{\left[\ \mathbf{p}_{\mathrm{WP_x}} - \ \mathbf{p}_{\mathrm{WO_x}}\right] \sin\theta_{\mathrm{P}} - \left[\ \mathbf{p}_{\mathrm{WP_y}} - \ \mathbf{p}_{\mathrm{WO_y}}\right] \cos\theta_{\mathrm{P}}}{\boldsymbol{v}_{\mathrm{P_y}}}$$

In addition,

- $\mathbf{p}_{\mathrm{WP_x}}$ and $\mathbf{p}_{\mathrm{WP_y}}$ denote the x and y components of the position vector from the origin of the world frame **{W}** to the surrounding object, P, expressed in **{W}**, originally measured in the LIDAR frame **{L}**. Note that it is the potential field of object P that causes a cost to be incurred by an object, O, whose future trajectory is to be predicted. Thus, it is at the position of object O at which the cost, $U_{t\,P}$, due to the potential field of P is to be evaluated.
- $\boldsymbol{v}_{\mathrm{P_x}}$ and $\boldsymbol{v}_{\mathrm{P_y}}$ are the x and y components of the velocity that object P possesses expressed in **{W}**.
- θ_{P} denotes the heading angle of object P obtained from $\boldsymbol{v}_{\mathrm{P}}$.
- $\mathbf{p}_{\mathrm{WO_x}}$ and $\mathbf{p}_{\mathrm{WO_y}}$ denote the x and y components of the position vector from the origin of **{W}** to the object, O, whose trajectory is to be predicted and who experiences the potential field of object P.
- a and b are the scaling constants that affect the shape of the potential field.
- s: Square of euclidean distance given x and y components of a vector – defined as: $s(x,y) = x^2 + y^2$.

Note that this function represents the equation of an ellipse that is rotated by θ_{P} and has semi-major and semi-minor axis as $\boldsymbol{v}_{\mathrm{P_x}}$ and $\boldsymbol{v}_{\mathrm{P_y}}$ and with the cost (analogous to the amount of repulsion) as the 3^{rd} dimension.

To further understand the contributions of each element in the potential field a graphical visualization tool has been utilized and prepared to simulate the aforementioned potential field function. It can be accessed at: https://www.desmos.com/calculator/oxgj1lwbbn.

4.1.2 Agent-space Potential Fields

Potential fields that model agent-space interactions require information such as road boundaries and lane boundaries from the map. Given the intention to evaluate results from the proposed approach on the Waymo dataset, illustration of potential field generation from road and lane boundaries extracted from this dataset is presented. To do so, a series of coordinates representing road boundaries are obtained from the dataset and a binary driveable space-based potential field is generated. This means that driveable areas do not lead to an incurred cost and elsewhere there is a fixed cost that is incurred.

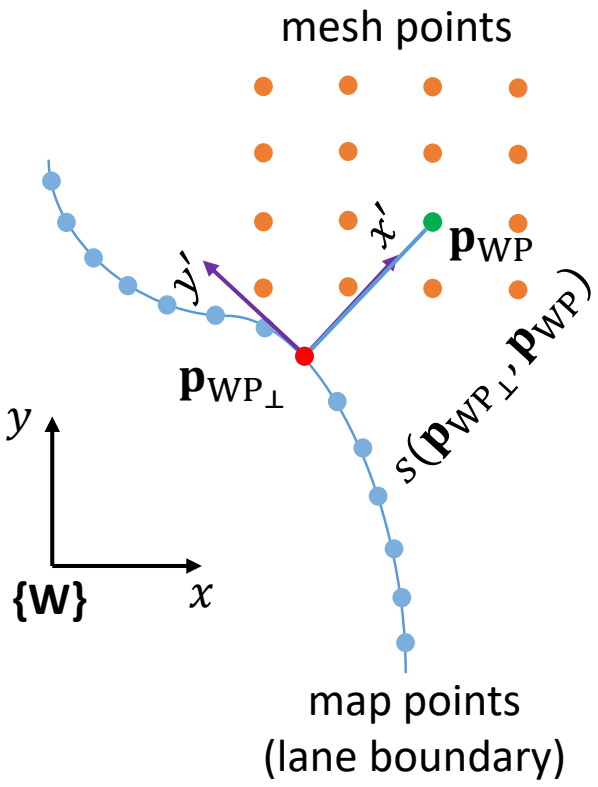


Figure 4.2: A visual illustration of lane boundary potential field generation involving terms in Equation (4.2).

For lane boundaries, a tolerance for being close to the edges of the lane instead of a binary mask must be modelled given that lane changes can occur often. This can be modelled via applying an exponential decay to the potential field as the distance from the

lane boundary coordinate of interest to that of the point where it is evaluated increases. Given that the potential field is evaluated at discrete mesh points, the orthogonal distance from the mesh point of interest to the nearest orthogonal can be used. A visual illustration of this is shown in Figure 4.2. This is modelled via Equation (4.2) below.

$$U_{t_{\text{Lane}}}\left(\mathbf{p}_{\text{WP}_\perp}, \mathbf{p}_{\text{WP}}, c, d\right) = ce^{-ds(\mathbf{p}_{\text{WP}_\perp}, \mathbf{p}_{\text{WP}})} \tag{4.2}$$

where,

- $\mathbf{p}_{\text{WP}}$ denotes the position vector of a point $\mathbf{p}$ originating from $\mathbf{\{W\}}$ at which the potential field is to be evaluated.
- $\mathbf{p}_{\text{WP}_\perp}$ denotes the position vector originating from $\mathbf{\{W\}}$ to a point that lies on the lane boundary from the map and is orthogonal to point $\mathbf{p}$.
- c and d are the scaling constants that affect the shape of the potential field.
- s: Square of euclidean distance given x and y components of a vector – defined as: $s(x, y) = x^2 + y^2$.

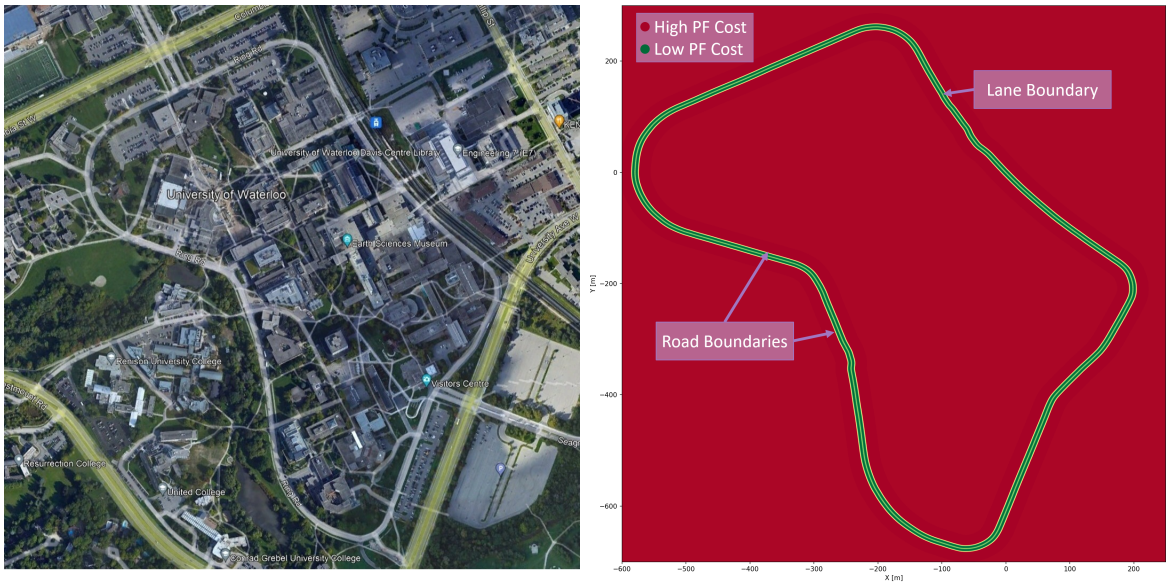
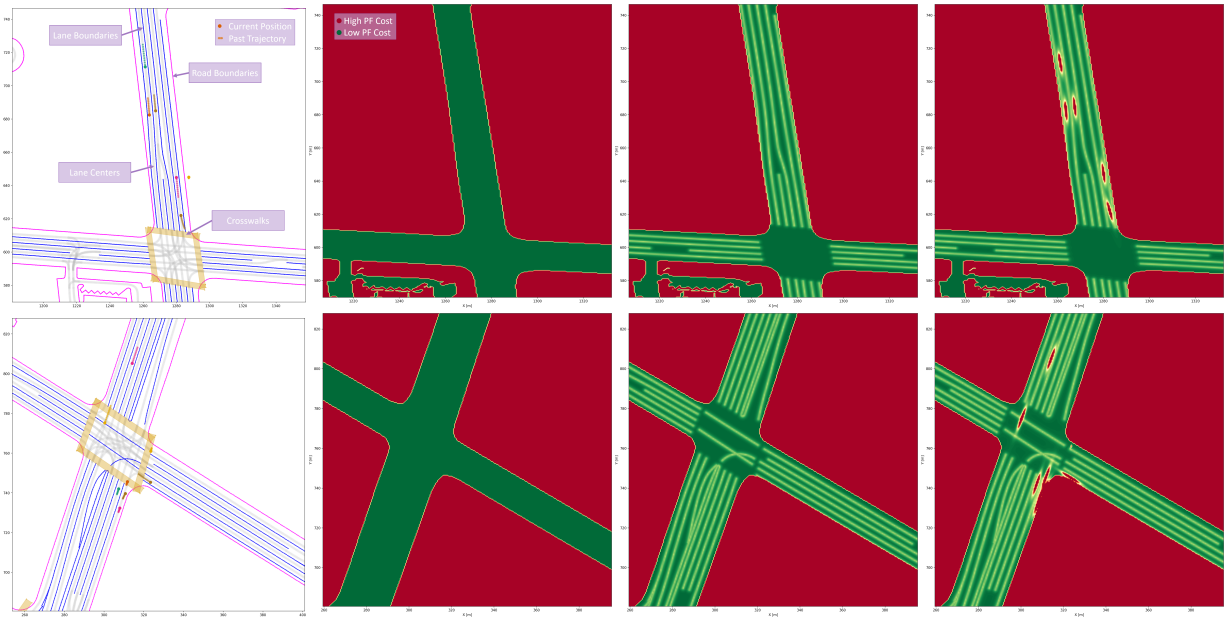


Figure 4.3: Illustration of the potential field for road and lane boundaries for Ring Road at University of Waterloo.



(a) Extracted map. (b) Road boundary PF. (c) Lane boundary PFs. (d) Object PFs.

Figure 4.4: Illustration of map information extracted from Waymo dataset for two scenarios and the sequential generation of potential fields due to road boundaries, lane boundaries, and objects respectively.

To illustrate generation of potential fields for both road and lane boundaries, an illustration of potential field generation for Ring Road at the University of Waterloo is first presented in Figure 4.3, followed by two scenarios from the dataset shown in Figure 4.4.

Ring Road is a 2.7 km circular two lane road with no physical divider in between the two lanes. The generated potential field based on the road boundaries is shown with red and green indicating high and low cost respectively. Note that there is also a lane boundary potential field generated at the lane marking dividing the two lanes. The potential field is not as high as the road boundaries give that vehicles can temporarily cross the lane marking if required. Further illustrations for both datasets are presented in Appendix B.

For Waymo dataset, the road boundaries, lane boundaries, lane centers (shown in magenta, blue, and grey), and object positions are extracted from the dataset and visualized in Figure 4.4a. The generated potential field based on the road boundaries is shown in Figure 4.4b with red and green indicating high and low cost respectively. Moreover, the lane boundary and object potential fields are shown in Figures 4.4c and 4.4d respectively. The generation of these potential fields will be utilized in the next subsection.

4.2 MPC-PF Trajectory Prediction

4.2.1 System Model

Given the potential field for all objects surrounding a given object whose trajectory is to be predicted, the subsequent task is to utilize this potential field to obtain socially informed trajectory predictions.

A key observation is that the primary state variable that influences the trajectory traversed by any object in general is the velocity – specifically the heading angle component. Thus, it is the **heading angle** of the object at each timestep in the future, $\boldsymbol{\Theta} = [\theta_{t+1}, \cdots, \theta_{t+h}]^\top$, that is to be predicted.

In addition, an object's speed (magnitude of velocity) remains nearly constant during the short time horizon, h, for which the prediction is generated, especially for vehicles. Thus, it is assumed that the object's speed follows a constant speed model while generating socially aware predictions – for instance while anticipating potential collisions with other objects. This can be considered as a valid assumption especially since the update rate of the position and velocity estimation module (such as MHE in [3]) is generally high and since the heading angle of the object encompasses social interactions via the potential field (embeds object speed inherently) which also contains map information. Thus, the inputs to this module from MHE for a given object are $\boldsymbol{y}_t$,$\boldsymbol{v}_t$, and θ_t. For object O:

$$\boldsymbol{y}_t = \begin{bmatrix} \mathbf{p}_{\mathrm{WO_x}} \\ \mathbf{p}_{\mathrm{WO_y}} \end{bmatrix}, \boldsymbol{v}_t = \begin{bmatrix} \boldsymbol{v}_{\mathrm{O_x}} \\ \boldsymbol{v}_{\mathrm{O_y}} \end{bmatrix}, \theta_t = \tan^{-1}\left(\frac{\boldsymbol{v}_{\mathrm{O_y}}}{\boldsymbol{v}_{\mathrm{O_x}}}\right) \tag{4.3}$$

Utilizing the constant speed assumption, the velocity update equation depends on the heading angle at a given timestep k. Note that this system model is also a discretized representation of the continuous time model through the constant velocity assumption between the discrete timesteps. Hence the velocity update is given by:

$$\boldsymbol{v}_{t+k} = \begin{bmatrix} |\boldsymbol{v}_t| \ \cos(\theta_{t+k}) \\ |\boldsymbol{v}_t| \ \sin(\theta_{t+k}) \end{bmatrix} \tag{4.4}$$

In general, the heading-based predicted position of an object (here velocity is not constant as heading angle changes, however, speed is constant) is computed as:

$$\boldsymbol{y}_{t+k} = \boldsymbol{y}_{t+k-1} + \boldsymbol{v}_{t+k}\Delta t \tag{4.5}$$

Note that if it is assumed that both the heading and speed remain constant (constant velocity), the reference trajectory is obtained by:

$$\boldsymbol{y}_{rt+k} = \boldsymbol{y}_t + \boldsymbol{v}_t\,[k\Delta t] \tag{4.6}$$

Remark: Given pedestrians do not track lane lines like vehicles, this constant velocity reference trajectory can be used as a base case trajectory with potential field refinements as elaborated in the next section. However, for vehicles the reference trajectory, $\boldsymbol{y}_{rt+k}$, can be obtained directly by constant speed-based querying of waypoint headings belonging to the current vehicle lane through Equations (4.4) and (4.5).

4.2.2 Design of MPC-PF Cost Function

The idea presented in this subsection is primarily focused on the observation that objects generally change their heading to track lane lines or follow environmental cues such as crosswalks, but otherwise do not change their heading unless there happens to be another object in their path. In addition, the heading of object is not expected to change drastically during a short period of time. Taking these key observations into account, the task is to predict future heading angles, $\boldsymbol{\Theta} = [\theta_{t+1}, \cdots, \theta_{t+h}]^\top$, for a future time horizon of length h.

Given the system model from the previous subsection, a cost function is designed to incorporate these primary observations. The objective is to minimize the following non-convex cost function with the minimization variable $\boldsymbol{\Theta}$:

$$\begin{aligned} \boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} \sum_{k=1}^{h} C_{ref} + C_{heading} + C_{PF} \qquad (4.7)\\ s.t. \quad \boldsymbol{y}_{t+k} = \boldsymbol{y}_{t+k-1} + \boldsymbol{v}_{t+k}\Delta t, \quad k = 1,\ \ldots,\ h \\ \boldsymbol{\Theta} = [\theta_{t+1}, \ldots, \ldots, \theta_{t+h}]^\top \\ -\pi/2 \leq \theta_{t+k} - \theta_t \leq \pi/2 \end{aligned}$$

where,

$$C_{ref} = \overbrace{\left\|\boldsymbol{y}_{t+k} - \boldsymbol{y}_{rt+k}\right\|_Q^2}^{\text{Reference Deviation}} \tag{4.8}$$

$$C_{heading} = \overbrace{r\left[\theta_{t+k} - \theta_{t+k-1}\right]^2}^{\text{Heading Deviation}} \tag{4.9}$$

$$C_{PF} = \overbrace{s[U_{t+k\,P_i}]}^{\text{PF Cost}} \tag{4.10}$$

and where,

- $\boldsymbol{y}_{t+k}$ and $\boldsymbol{y}_{rt+k}$ denote the predicted and reference position vectors of the object of interest respectively at timestep k expressed in $\mathbf{\{W\}}$ computed as per Equations (4.5) and (4.6) above.
- θ_t is the current heading angle obtained through the MHE scheme while θ_{t+k-1} and θ_{t+k} represent the heading angle of the object at timestep $t+k-1$ and $t+k$.
- $U_{t+k\,P_i}$ is the accumulated cost due to the potential field all objects, P_i, surrounding the object of interest, O, as well as due to map cues and is computed as shown in Equation (4.1). This term is a function of multiple inputs including positions of any two given objects, velocity, and heading.
- Q is a scalar matrix while r and s are scalar constants acting as weights for the three terms in the cost function.

Reference Deviation: The first term in the cost function, referred to as reference deviation and defined in Equation (4.8), incorporates the observation that objects generally tend to continue moving with the same speed in absence of any obstacles in their path, at least for a short period of time. However, the heading may change due to the need to track a lane line for instance or due to presence of another object. Thus, the predicted trajectory, derived from the primary prediction variable $\mathbf{\Theta}$ through Equation (4.7), should be such that it matches the reference trajectory (constant velocity for pedestrians and based on constant speed lane line tracking for vehicles), $\boldsymbol{y}_{rt+k}$, as shown in Figure 4.5. This is precisely the goal that the first term in the cost function aims to accomplish.

Heading Deviation: The second term in the cost function, referred to as heading deviation or the proximity term and defined in Equation (4.9), addresses the observation that any object's heading does not change instantaneously. This is done by minimizing the difference between any two given heading predictions belonging to adjacent timesteps. In addition, the constraint imposed on the *difference* between the current heading estimate from the MHE scheme and any given heading prediction within the prediction horizon,

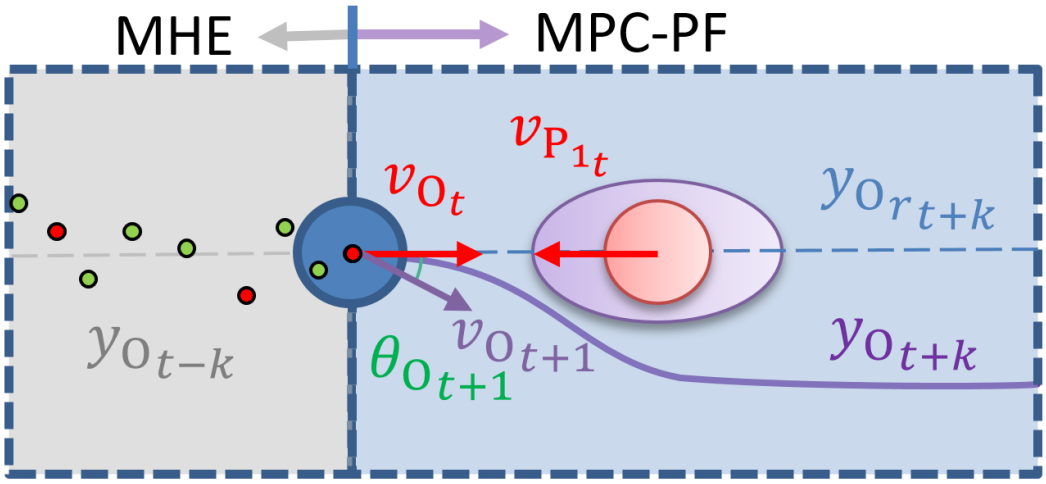


Figure 4.5: Illustration of the potential field, reference trajectory, and corresponding trajectory for a simple two pedestrian scenario.

$\theta_{t+k} - \theta_t$, bounds it between $[-\pi/2, \pi/2]$ to aid in narrowing down the global minima during optimization.

Potential Field Cost: Finally, the third term in the cost function, referred to as PF Cost and defined in Equation (4.10), enables incorporating social interactions through the potential field cost formulated in Equation (4.1). Note that $U_{t+k\,P_i}$ is the accumulated cost due to the potential field of all objects surrounding the object of interest, O, where the input $\mathbf{p}_{\text{WO}}$ is replaced by $\boldsymbol{y}_{t+k}$. Thus, during the minimization, the optimal solution for the predicted headings will be one that minimizes the total cost incurred throughout the prediction horizon due to being close to surrounding objects. In addition, road boundaries have a group of steep potential fields situated along the bounding lines whereas lanes that can be crossed over have a potential field with the cost capped at a low nominal value. Pedestrian crosswalks are modelled similar to these lanes that have the potential field cost capped.

Overall, the tradeoff between the deviation from the constant velocity trajectory and the cost due to being within the potential field of surrounding objects while ensuring gradual changes in the predicted headings enables sound trajectory predictions that include temporal information as well as map cues and consider social interactions amongst objects justifying achievement of the three key points of contribution.

4.2.3 Cost Function Optimization

Given the optimization problem above is non-linear and non-convex, it is proposed that the Sequential Least Squared Quadratic Programming (SLSQP), a Sequential Quadratic Programming (SQP)-based optimization algorithm proposed in [71], be used to obtain the solution to the optimization problem.

SQP is an iterative method for non-linear optimization problems where the cost function and the constraints are twice continuously differentiable. The SLSQP optimizer iteratively solves multiple QP subproblems utilizing the Han–Powell Quasi–Newton method with a Broyden–Fletcher–Goldfarb–Shanno (BFGS) update [72, 73, 74, 75].

The optimizer uses a slightly modified version of Lawson and Hanson's Non-Negative Least-Squares (NNLS) solver [76, 77]. As a whole, SLSQP, can be utilized for solving nonlinear optimization problems that minimize a scalar function in the form:

$$\boldsymbol{x}^{*} = \arg\min_{\boldsymbol{x}} f\left(\boldsymbol{x}\right); f\left(\boldsymbol{x}\right) \epsilon\ \mathbb{R}^{n} \tag{4.11}$$

$$\text{s.t. } h_j\left(\boldsymbol{x}\right) = 0; \quad j = 1, \ldots, n_k$$

$$g_j\left(\boldsymbol{x}\right) \leq 0; \quad j = n_{k+1}, \ \ldots, m$$

$$\text{where, } L \leq \boldsymbol{x} \leq U$$

Here, $f\left(\boldsymbol{x}\right)$ is the cost function and $h_j\left(\boldsymbol{x}\right)$ and $g_j\left(\boldsymbol{x}\right)$ denote the equality and inequality constraint functions respectively and L and U are lower and upper bounds on the state variables. For the problem at hand, the first two terms in Equation (4.7) are quadratic and convex. However, the potential field cost term: $s[U_{t+k\,P_i}]$ is continuously differentiable everywhere except at the position of the surrounding objects since the denominator term in Equation (4.1) becomes zero when evaluated at the position of the object of interest itself or if the object velocity components are exactly zero.

Since, the cost function shown above is not twice continuously differentiable, as required by the SQP problem, the problem is ill-formed without any stability checks. The second derivative of the cost function with respect to the minimization variables contained in $\boldsymbol{\Theta}$, defined in Equation (4.7), not only depends on a given timestep $t+k$, but also on the prior timesteps.

For brevity, the second derivative has not been shown here, however, it is evident that the discontinuity of the second derivative arises in the case when $\mathbf{p}_{\mathrm{WP_x}} = \mathbf{p}_{\mathrm{WO_x}}$

Algorithm 3: MPC-PF: Trajectory Prediction

Input: Agent positions: ${}^{\mathrm{W}}\mathbf{p}_{\mathrm{WP_i}}$, velocity: ${}^{\mathrm{W}}\boldsymbol{v}_{\mathrm{P}_i}$, HD map: lane centers and crosswalks all at time t

Output: Predicted Trajectories: ${}^{\mathrm{W}}\boldsymbol{y}_{t+k\mathrm{P_i}}, k\,\epsilon\,[1, h]$

Initialize: $h, \Delta t, a, b, Q, r, s, \boldsymbol{\Theta} = [\theta_t]$

for *agent* O ***in*** *agents* P_i **do**

  Obtain $\boldsymbol{y}_t$, $\boldsymbol{v}_t$, θ_t from Eq. (4.3) and call SLSQP:

  while $\boldsymbol{\Theta} \neq \boldsymbol{\Theta}^*$ *or* $i < i_{max}$ **do**

    1. Compute $\boldsymbol{v}_{t+k}$, $\boldsymbol{y}_{t+k}$ using Eq. (4.4) and (4.5)
    2. Compute $\boldsymbol{y}_{rt+k}$ from Eq. (4.6)
    3. Compute $U_{t+kP_i}\left(\mathbf{p}_{\mathrm{WP}_i}, \mathbf{p}_{\mathrm{WO}}, \boldsymbol{v}_{\mathrm{P}_i}, a, b\right)$ $\forall\, \mathrm{P}_i$ and HD map cues from Eq. (4.1)
    4. Using $\boldsymbol{y}_{t+k}, \boldsymbol{y}_{rt+k}, \boldsymbol{\Theta}$, U_{t+kP_i} obtain $C_{ref}, C_{heading}, C_{PF}$
    5. SLSQP update of $\boldsymbol{\Theta}$ based on cost in Eq. (4.7)

  end

end

and $\mathbf{p}_{\mathrm{WP_y}} = \mathbf{p}_{\mathrm{WO_y}}$ which is when the object has collided with another surrounding object in the environment or when $\boldsymbol{v}_{\mathrm{P_x}}$ and $\boldsymbol{v}_{\mathrm{P_y}} = 0$. Although, practically this is not anticipated to occur, this must be dealt with for stability of the solver. Given that this is the only case when the derivative and second derivative of the cost function becomes discontinuous, it suffices to add a small negligible constant ϵ to these terms in Equation (4.1).

This ensures that the denominator remains non-zero and mitigates the issue. Note that the inequality constraints are directly imposed on the minimization variable $\boldsymbol{\Theta}$, defined in Equation (4.7), and hence is not a composite function of itself. Thus, the constraints remain twice differentiable. Thus, the new cost function is now well-formed for the SQP-based minimization problem and thus the SLSQP method can be used for the problem at hand with stability checks in the implementation of the algorithm. The overall MPC-PF approach is outlined in Algorithm 3.

It is important to note that the optimization problem has been heuristically constrained into a significantly smaller search space by limiting the predicted heading angle, θ_{t+k}, to fall within $-\pi/2$ and $\pi/2$ of the previously obtained heading angle, θ_{t+k-1}. This, with the combination of potential field information from road boundaries, crosswalks, and other objects mitigates undesired convergence to a local minimum that is spatially far from the global solution.

4.3 Summary

In this chapter, MPC-PF, a novel potential field-based trajectory predictor that incorporates social interaction via agent-agent and agent-space considerations was introduced. MPC-PF takes position and velocity estimates from MHE as inputs and is able to tradeoff between inherent model biases across the prediction horizon yielding trajectory predictions as an output. To do so, the models for agent-agent potential fields as well as that for agent-space potential fields were introduced. This included map features such as road and lane boundaries. The approach utilizes a cost function, as expressed in Equation (4.7), incorporating cost for deviation from the reference path, abrupt change in heading, and cost incurred due to being close to other objects or map features via potential fields ultimately capturing social interaction. Furthermore, details on obtaining the solution to the non-linear and non-convex cost function via SLSQP were also outlined. The overall algorithm is summarized in Algorithm 3.

Chapter 5

RL-PF: Potential Field-Based Deep Deterministic Policy Gradient Agent for Trajectory Prediction

In this chapter, a DDPG trajectory prediction agent, termed RL-PF, is presented and elaborated upon. In Section 5.1, first the action space and state space are defined in Section 5.1.1 and 5.1.2 respectively. Subsequently, important reward function design considerations are outlined to ensure faster model convergence and adequate exploration. The reward function is defined in Section 5.1.3. Thereafter, in Section 5.1.4, the network architecture is depicted. Furthermore, in Section 5.2 the performance of the DDPG agent is evaluated. Specifically, first the training environment setup is established in Section 5.2.1 followed by discussion on the critic and actor losses along with the average reward across training episodes in Section 5.2.2. Most importantly, qualitative evaluation of the trained agent on two primary scenarios of interest is conducted in Section 5.2.3. Lastly, few conclusions on the performance of the approach are presented in Section 5.3.

5.1 Formulation and Approach

The process to predict an object's motion can be modelled as a Markov Decision Process (MDP) [78]. A MDP consists of: (1) a continuous state space, S, (2) a continuous action space, A, an (3) unknown transition, T, and a (4) reward function, R. Once such MDP is show in Figure 5.1. At each state the action taken by the agent depends on the current

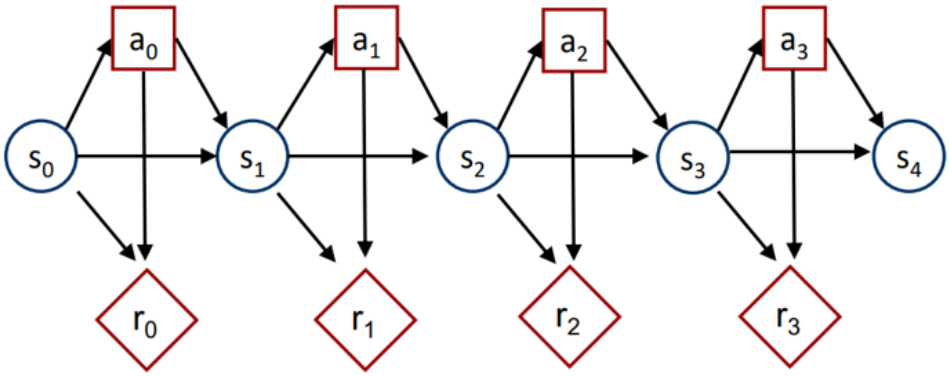


Figure 5.1: Illustration of the dependence of state, action, and reward in a MDP.

state, previous states, and the sum of cumulative reward. The goal of the proposed approach is to learn a policy $\boldsymbol{\pi}$ while maximizing the cumulative reward and incorporating uncertainty.

5.1.1 Action Space

The action space for any object at a given time, t, consists of the heading angle, θ_t, and speed, $\|\boldsymbol{v}_t\|$. The combination of the heading angle with speed leads to planar motion of the object. As discussed previously, the heading angle of an object primarily governs social interaction and affects the trajectory traversed by any object. The heading angle of an object is a continuous quantity and hence the proposed approach must be compatible with this continuous action space.

The speed of any object generally does not change drastically. Assuming that the speed of an object remains nearly constant during a short and finite prediction horizon is generally valid, especially at urban intersections. It is rather important to model the interaction between objects which can be done based on the current heading angle and speed of the object. Thus, the object speed does not need to be included in the action space.

Thus, the action space consists of the heading angle of the object expressed in degrees as: $A = [\theta]$. In addition, the heading angle for both vehicles and pedestrians can be constrained in the range $\theta \in$ [-45,45] degrees to account for physical limitations and to act as a termination condition for training speedup.

5.1.2 State Space

The state is observed directly from the environment and is to be crafted carefully to include observations that do not depend on the absolute values of the observations pertaining to the training environment to allow for generalization. In addition, given that the policy learnt is a mapping from the state to action (i.e. heading angle), $\boldsymbol{\pi}(s) \rightarrow a$ where $s \in S$ and $a \in A$, it is critical to include observations that encode the necessary physical factors that influence selection of a probable action.

Thus, the state must not only include the heading angle of the object of interest, but also a measure of proximity to surrounding objects and road features. Potential field provide a good indication of proximity to surrounding object and thus it can be included in the state. Thus, $S = [\theta, U_{P_1}, ..., U_{P_n}]$, where n is the total number of objects and road features. Note that the amount of relevant objects around the object of interest is clipped to 15 as empirically it was found that this models the interaction influence well.

5.1.3 Designing a Reward Function

The design of the reward function is arguably the most important task that directly influences the learning performance of the agent. There are three primary design considerations while constructing the reward function. Specifically,

- Avoiding sparsity in the reward function to ensure faster convergence. Generally, a continuous and high frequency reward function is preferred.
- Inclusion of positive rewards help encourage greedy exploration. This must be complimented with a termination condition otherwise exploration will prevail for entire episode length and exploitation will not occur.
- Inclusion of negative reward aids in faster convergence to a terminal state. This must be complimented with positive rewards otherwise the episode will end abruptly.

Moreover, for motion prediction, there are three primary desired attributes that the designed reward function should include: (1) **Encourage maintaining** the same heading angle for the prediction horizon if no other objects are present in proximity of the direction of movement, (2) **Discourage being in close proximity** to other objects if present, and (3) **Maximize episode duration** to prevent immature termination.

The integration of the potential field-based cost into the reward model is a plausible way to impose "soft" rules for collision avoidance and social interaction. Consequently, the reward function is split into three main parts: (a) Reward due to deviation from reference position predicted by a constant velocity model, (b) Potential field-based reward due to both surrounding objects and map features such as lane or crosswalk boundaries, and (c) Reward due to longer episode duration. Specifically, the reward function is expressed in Equation (5.1).

$$R = R_{start} + R_{dev} + R_{PF} + R_{duration} + R_{eterm} \tag{5.1}$$

where,

$$R_{start} = +w_{start} \tag{5.2}$$

$$R_{dev} = -w_{dev} \sum_{k=1}^{h} \left\| \boldsymbol{y}_{t+k} - \boldsymbol{y}_{rt+k} \right\|, \boldsymbol{\pi}(s_t) \rightarrow \theta_t \rightarrow \boldsymbol{y}_{t+k} \tag{5.3}$$

$$R_{PF} = -w_{PF} \sum_{i=1}^{n} U_{t+k P_i} \tag{5.4}$$

$$R_{duration} = +w_{duration}\, m \tag{5.5}$$

$$R_{eterm} = -w_{eterm} \tag{5.6}$$

The first term in the reward function expressed in Equation (5.2) is a default positive reward term to ensure that the total reward remains net positive per the aforementioned design considerations. The next term expressed in Equation (5.3) ensures that the agent continues to traverse a trajectory that is in line with the heading angle at a given timestep by penalizing deviation in position from the reference constant velocity trajectory. Note that $\boldsymbol{y}_{t+k}$ and $\boldsymbol{y}_{rt+k}$ were defined previous in Equation (4.5) and (4.6) respectively. Most importantly the potential field-based reward expressed in Equation (5.4) encodes the information necessary for the object to maximize the distance between the surrounding objects and embeds map feature potential fields. For instance, as the distance between the object of interest and any other object reduces, a higher negative reward is realized which intuitively incentivizes the agent to remain away from other objects. Furthermore, the duration-based reward term expressed in Equation (5.5) provides an incentive for the agent to avoid reaching a terminal state by providing a positive reward proportional to the timestep. Lastly, a constant negative reward term, as expressed in Equation (5.6), ensures faster convergence to the optimal policy.

5.1.4 DDPG Agent For Trajectory Prediction

DDPG is a model-free off-policy actor critic algorithm that was first proposed in [79]. Unlike, Deep Q-Networks (DQN) and other actor-critic networks, the primary advantage of DDPG is its compatibility with continuous action spaces [80] [81]. This is the one of the main reasons for its suitability to the problem at hand. DDPG utilizes four networks in total: actor μ, actor target μ', critic Q, and a critic target network Q'. In DQN, due to the presence of a discrete input, the optimal policy can be obtained via Equation (5.7).

$$\boldsymbol{\pi}^* = \arg\max_a Q(s, a) \tag{5.7}$$

In DDPG, the actor is policy network which is trained to yield the optimal action given the state as an input and thus the argmax operator can be bypassed. The actor policy network is updated using the observed samples (replaces expectation) using the **deterministic** (as opposed to stochastic) policy gradient as follows:

$$J_\beta(\theta) = \mathbb{E}_{s\sim\rho^\beta}[\nabla_a Q^\mu(s, a)\nabla_\theta \mu_\theta(s)|_{a=\mu_\theta(s)}] \tag{5.8}$$

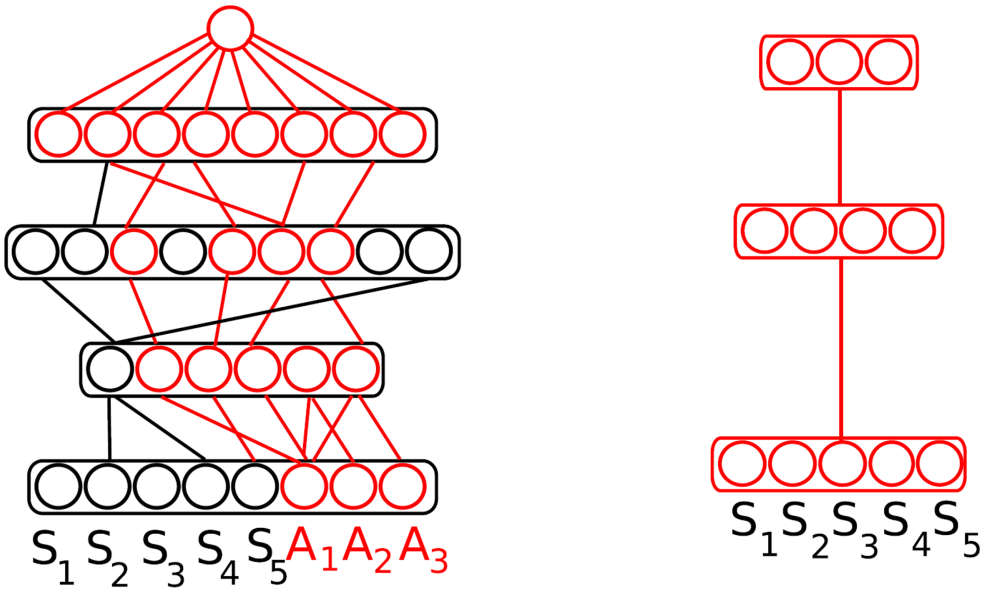


Figure 5.2: The DDPG network architecture consists of an actor and critic network along with their corresponding target networks.

The overall network schematic is shown in Figure 5.2 and an overview of the vanilla DDPG algorithm is presented in Algorithm 4.

Algorithm 4: RL-PF: DDPG Agent

Input: Current state: s
Output: Trained policy: $\boldsymbol{\pi}_\theta$
Initialize: actor network μ yielding initial $\boldsymbol{\pi}_\theta$
for m *episodes* **do**
  Initialize s_0 and set $n \leftarrow 0$
  while s *is not terminal* **do**
    Select $a_n = \boldsymbol{\pi}_\theta(s_n)$
    Execute a_n, observe s_{n+1}, r_n
    $\delta \leftarrow r_n + \gamma Q_w(s_{n+1}, \boldsymbol{\pi}_\theta(s_{n+1})) - Q_w(s_n, a_n)$
    Update Q: $w \leftarrow w + \alpha_w \gamma^n \delta \nabla_w Q_w(s_n, a_n)$
    Update $\boldsymbol{\pi}$: $\theta \leftarrow \theta + \alpha_\theta \gamma^n \nabla_\theta \boldsymbol{\pi}_\theta(s_n) \nabla_a Q_w(s_n, a_n)|_{a_n = \boldsymbol{\pi}_\theta(s_n)}$
    $n \leftarrow n + 1$
  end
end

Upon training, the actor network μ yields a policy that outputs the heading angle that best models the three considerations in the reward function, $\boldsymbol{\pi}^*(s_t) \rightarrow {\theta_t}^*$. Note that given this policy, at any given time t, the heading angle can be computed from the policy and the trajectory of surrounding objects, $\boldsymbol{y}_{t+k}$, can be obtained via a sequential rollout as presented in Equation (4.5).

The training of the DDPG agent includes sequential update of the critic and the actor as per Equation (5.8). However, since the actor maps the state features to actions which in turn correspond to a continuous Q function output value, the critic input space is appended with the action vector allowing joint training using stochastic gradient descent. For agent training, a 2 layer fully connected critic and actor network with **400** nodes for the first hidden layer, **300** nodes for the second hidden layer, and **ReLu** activations were used [82].

Figure 5.3: CARLA Unreal Engine simulation environment with macad-gym environment interface used for training.

5.2 RL-PF - Results and Discussion

5.2.1 Training Environment Setup

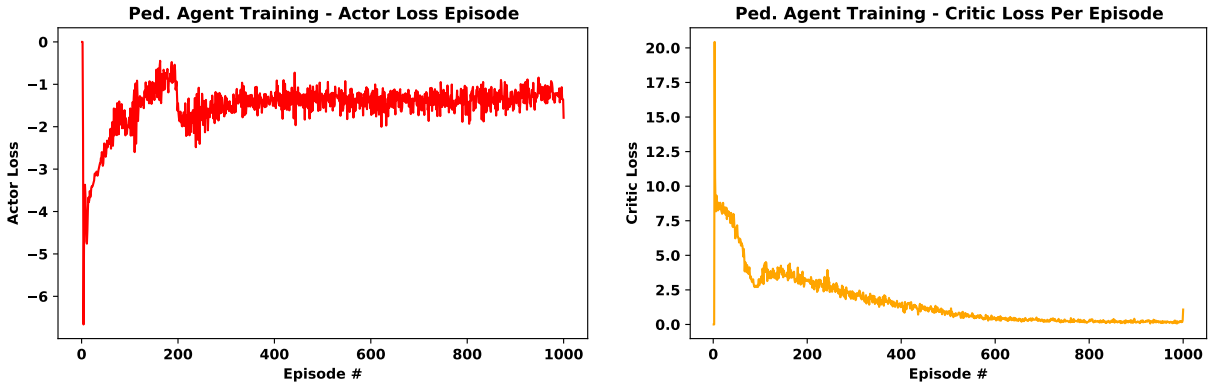
The DDPG agent training was conducted using the CARLA project in Unreal Engine 4.22 [83]. To enable future multi-agent training and ease of access to observations and agent control, the macad-gym project developed by [84] was used. It offers an OpenAI gym-based interface and is convenient for training and testing purposes. A sample illustration of the training environment is shown in Figure 5.3 and the corresponding video can be found at https://youtu.be/WOcei54XW-U.

The training scenarios consist of several urban intersections with multiple pedestrians attempting to cross the street as shown in Figure 5.3.

5.2.2 Training Details

A soft update for both actor and critic network was used with $\tau = \mathbf{0.01}$. Furthermore, a learning rate of 0.0001 for the actor network and 0.001 for the critic network was used. A replay buffer of size 10000 and batch size of 128 was also employed with γ=0.99. The agent was trained for $m = 1500$ episodes with an episode step duration cutoff of 100 steps (0.05 seconds of simulation time per episode). The agent was trained on a NVIDIA RTX 2080 GPU with an Intel i9-9900K processor and 32GB RAM. The total training time was nearly 6 hours.

Upon training the DDPG-based agent, the agent was able to learn a policy that intuitively tackles the aforementioned desired attributes. Figures 5.4a and 5.4b depict the



(a) (b)

Figure 5.4: Both the actor and critic losses converge after around 1000 episodes.

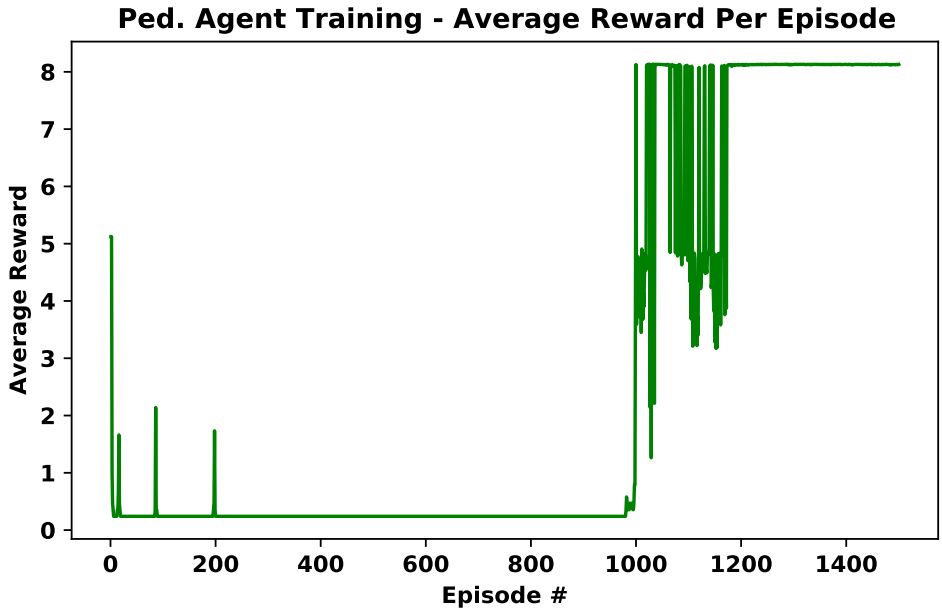


Figure 5.5: Average reward per episode converges after about 1200 episodes.

actor and critic losses and Figure 5.5 depicts the average reward per episode. As in Figure 5.4, both the actor and critic network losses seem to have converged (actor loss has not converged directly to 0) and from Figure 5.5, it is evident that the average reward has

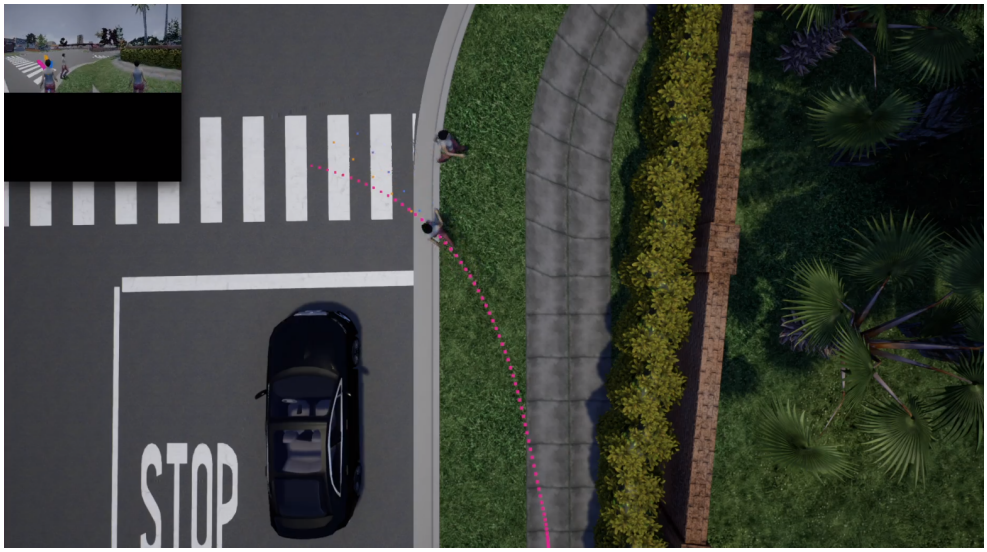


Figure 5.6: Scenario 1 - two pedestrians are approaching each other. Predicting the intent of the pedestrian that is entering enables safe decision making for the black vehicle at the stop sign.

stabilized and is indeed the optimal reward that can be achieved. These figures confirm that the agent has learned the optimal policy and confirms the capability of the RL-PF prediction scheme.

5.2.3 Qualitative Evaluation

The DDPG agent is evaluated on two primary scenarios of interest - Scenario 1: Anticipating intent at a stop sign controlled intersection and Scenario 2: Navigating multiple pedestrians at a crosswalk.

In Scenario 1, two pedestrians approach each other wherein one pedestrian is about to leave the crosswalk while the other is just about to enter as shown in Figure 5.6. The vehicle in black arrives at the stop sign controlled intersection and is assumed to be the SDV for which a decision to proceed or not needs to be made as is the case in typical autonomous driving encounters at urban settings. In doing so, predicting the intent of the pedestrian entering the crosswalk is critical and essential for safe autonomous operation. Decision making merely based on presence of pedestrians within or near the crosswalk is not sufficient; rather, anticipation of their intent is required. Both pedestrians are stationary

at the beginning and reach their maximum velocity of 1.8 m/sec. This is generally the maximum velocity that can be attained by a pedestrian while walking.

In Figure 5.6, the predicted trajectory of the pedestrian entering the crosswalk is shown in orange along with the constant velocity trajectory in blue and the actual traversed trajectory in magenta. The potential field for the crosswalk boundaries is essential for the prediction algorithm without which the predicted trajectory would coincide with the constant velocity trajectory leading the predicted positions to be outside the crosswalk in cases where the heading of the pedestrian is not aligned with the crosswalk, especially near the boundaries of the crosswalk.

Initially, when the pedestrians are not in vicinity of each other, the predicted trajectory coincides with the constant velocity trajectory as minimal negative reward due to reference path deviation and PF cost is incurred. When the pedestrian of interest is close to the pedestrian leaving the crosswalk, the predicted trajectory starts to deviate from the constant velocity trajectory and approaches the actual trajectory due to the anticipated social interaction between the two pedestrians. The predicted trajectory is generated with the anticipation that the other pedestrian will continue to traverse at a constant velocity and hence closer to the end of the horizon, it is anticipated that the other pedestrian's position would have moved closer towards the pedestrian of interest. This shows that the inherent tradeoff considered between the reward for reference deviation and the PF cost that the policy has learned.

Moreover, as the pedestrian leaving the crosswalk turns away from the pedestrian of interest and as the pedestrian entering the crosswalk approaches toward the boundary of the crosswalk, the potential field corresponding to the upper crosswalk boundary prevents the predicted trajectory to go out of the crosswalk, as expected. Thus, the results from this example confirm that the predicted trajectory for the pedestrian of interest captures the three primary observations about object motion as aforementioned in the design of the reward function This enables safe decision making for the vehicle in black through the computation of the anticipated time to collision within decision making. The corresponding video sequence of the trajectory visualization for this scenario is available at: https://youtu.be/UI_IzUIpNjg.

In Scenario 2, a typical multi-pedestrian encounter at a crosswalk is considered. There are three pedestrians are involved and the pedestrian of interest enters the crosswalk from the right side. Two pedestrians approach the pedestrian of interest in the opposite direction as shown in Figures 5.7a and 5.7b with all pedestrians stationary at the start and reaching maximum velocity of 1.8 m/sec similar to the previous example.

As shown in Figure 5.7a, the predicted trajectory deviates from the constant velocity

(a)

(b)

Figure 5.7: Scenario 2 - multiple pedestrians approach the pedestrian of interest at different timesteps. Each surrounding pedestrian is anticipated to collide with the pedestrian of interest.

trajectory and matches the actual trajectory due to an anticipated collision similar to the previous example. Subsequently, the predicted trajectory deviates away from the upper crosswalk boundary at first and then from the lower crosswalk boundary due to the potential field leading to a high negative PF reward as shown in Figure 5.7b. This example confirms the suitability of the approach in a dense multi-object environments and its applicability in predicting trajectory that models social interaction well.

Overall, based on the learned policy, the actor network can be used to map the state features observed at any given timestep to an optimal action and an internal update scheme based on the assumed forward velocity and policy derived steer angle can be used to generate the predicted position of the pedestrian of interest in a finite future horizon. The trajectory prediction scheme can run just over 30 frames per second and hence is capable of running in real-time. The corresponding video sequence of the trajectory visualization for this scenario is available at: https://youtu.be/7vRiWHxEOss.

5.3 Summary

In this chapter, the design of a DDPG-based RL-PF trajectory prediction framework was presented. The overall algorithm is summarized in Algorithm 4. The approach allows training with a continuous state and action space and the potential field-based heuristic reward function eliminates direct dependence on agent states resulting in better generalization. In developing the prediction model, it was noted that the design of the potential field-based heuristic was of major importance in developing a desirable reward function. The applicability of the approach in learning a policy that appropriately tracks the ground truth was confirmed by the experimental results indicating promising qualitative results. The results indicate that the three aforementioned primary desired attributes being: Encourage maintaining the same heading angle in absence other objects in proximity of the direction of movement, Discourage being in close proximity to other objects, and maximize episode duration to prevent immature termination were met. In addition, the approach is capable of running in real-time and the visualized results confirm that the designed reward function is suitable for the problem posed.

Chapter 6

Results and Discussions

In this chapter, a comprehensive evaluation of the performance of both MHE and MPC-PF is conducted. In Section 6.1, first the experimental setup to evaluate MHE is established in Section 6.1.1 followed by results from the fusion of 2D bounding box and LIDAR point cloud in 6.1.2 confirming real-time processing capability. Subsequently, in Section 6.1.3 and 6.1.4, the performance of the approach while the SDV is stationary or moving is presented. Thereafter, in Section 6.2, the evaluation setup is established for scenarios from the Waymo Dataset as well as other common scenarios at intersections. Furthermore, the qualitative and quantitative results are presented in Sections 6.2.2 and 6.2.4 respectively per the evaluation metrics defined in 6.2.3. Lastly, few conclusions on the performance of the approach are presented in Section 6.4 (see **video** at https://youtu.be/2ORorrOtIfE).

6.1 MHE: Experimental Results

To evaluate the performance of the proposed algorithm, two test scenarios were designed: estimation of the trajectory and states of a crossing pedestrian while the SDV is Case (1): stationary and Case (2): in motion. MHE parameters are presented in Table 6.1.

6.1.1 Experimental Setup

The measurements from the monocular camera, LIDAR, and the GPS+IMU form the necessary sensory inputs required by the MHE. In our experiments, the SDV depicted in Figure 6.1, which is the WATonoBus platform which the author of this thesis has been

Figure 6.1: Illustration of the WATonoBus platform equipped with cameras, LIDARs, GPS, computing, and an autonomous software stack used to perform experiments.

heavily working on and leading overall software developments since the beginning of the project, is instrumented with a 3.2MP Basler Ace camera, a combination of 32 channel Velodyne and Robosense LIDARs, and Applanix POS LVX to obtain these measurements respectively. The sensor dimensions, mounting positions, reference frames, and transforms are shown in Figure 6.2. Additional details on WATonoBus are presented in Appendix C.

Moreover, the algorithm was tested on a single machine with a NVIDIA RTX2060 (mobile) GPU, Intel i7-9750 Processor and 16 GB RAM (comparatively low end). The processing time attributed to the DNN object detector is **33** ms and that for the LIDAR Depth to image projection is less than **2** ms. The MHE scheme takes less than **55** ms to execute and can hence perform in real-time. Given the overview of the setup, the performance of the algorithm in cases (1) and (2) is discussed in the subsequent sections.

6.1.2 Bounding Box Informed Depth Projection

The bounding box informed depth projection and association search resulted in a significant reduction in computation time on the specified test hardware. In particular, over 15 times reduction in the computation time for the depth projection was observed. The 2D object

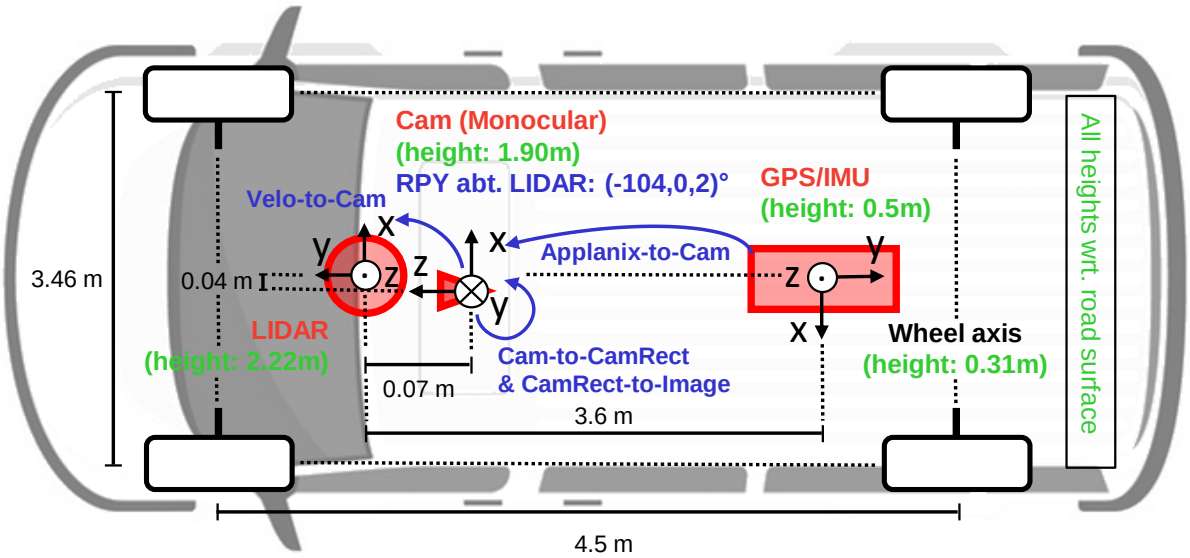


Figure 6.2: Illustration of position of the sensors (red) relative to the vehicle body and important distances (in meters). Heights above ground (green) are measured with respect to the road surface. Transformations between sensors are shown in blue.

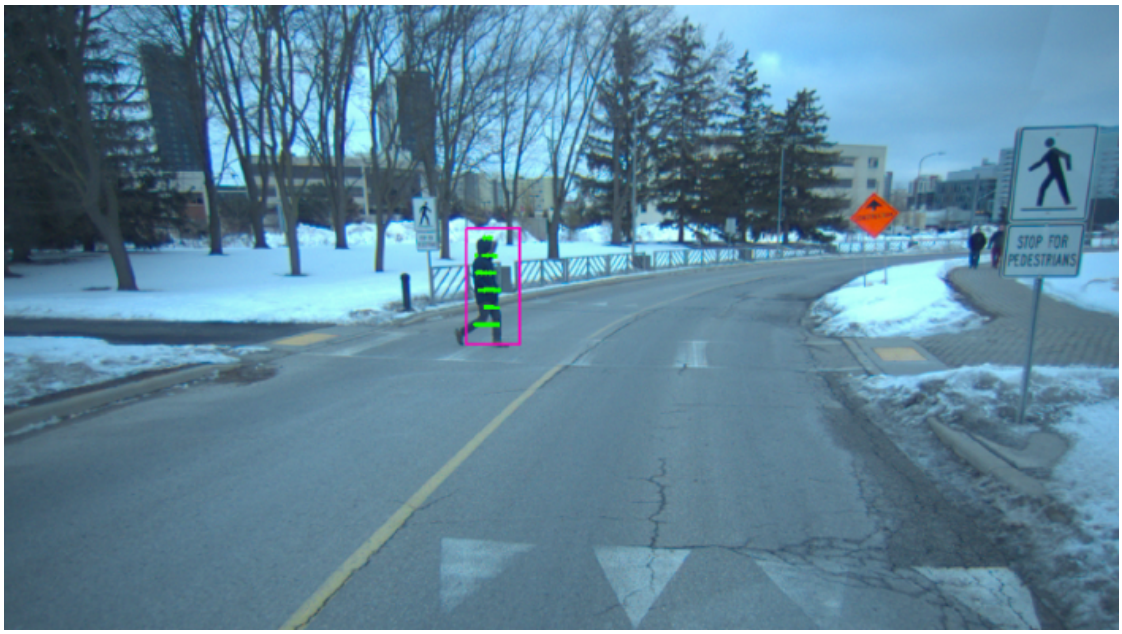
detector (YOLOv4 darknet [85]) runs at 30 fps and the LIDAR data frequency was 10 Hz. This leaves time room of close to 60ms for the MHE algorithm itself and our results indicate that the proposed algorithm can run without any bottlenecks. This is part due to the use of DNN libraries and implementation of the code using hardware specific libraries which is elaborated further in Appendix A. Sample results from the depth projection are shown in Figures 6.3a and 6.3b.

6.1.3 Experiment 1 - Stationary SDV

In this experiment, the SDV has come to a stop and the object under consideration, which is a pedestrian, crosses the street going from right to left in the image frame as shown in Figure 6.3a.

The pedestrian's estimated position at each timestep for the duration of the crossing act, is shown in Figure 6.4. Our approach estimates the pedestrian's trajectory with a maximum position tracking error of **50** cm. Moreover, due to formulation of the MHE in 3D with respect to the initial local reference frame being the GPS frame, the position of the pedestrian with respect to the WGS84 world frame can be easily obtained as shown

(a) Case (1) - Stopped SDV



(b) Case (2) - SDV with longitudinal and lateral motion

Figure 6.3: Illustration of detected object and depth projection. Both cases face intermittent detection loss and occlusion

Table 6.1: Parameters used for MHE Experiments.

Parameter	Value	Parameter	Value
K	$\begin{bmatrix} 1319.83 & 0 & 1068.32 \\ 0 & 1317.22 & 756.71 \\ 0 & 0 & 1 \end{bmatrix}$	Δt	0.1 s
V_j	$\begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	W	100 $\mathbf{I}_{6\times6}$
L	10	u, v	$\in$ [-2,2] m/s
a_x, a_y	$\in$ [-1,1] m/s^2	z	$0 \pm \epsilon$ m/s

in Figure 6.4. Moreover, a plot of the estimated centroid height of the pedestrian can also be obtained as shown in Figure 6.4, illustrating one of the primary advantages of this approach being formulated in 3D.

Since, in majority of the 2D detections, the bounding box is taller than the pedestrian itself, qualitatively the estimated height (derived from the bounding box centroid) is expected to be higher than half the height of the pedestrian (measured to be 1.8 m) and the results are in line with this.

Furthermore, utilizing the constrained inequalities for the states, the lateral and longitudinal velocity of the pedestrian is constrained between -2 to 2 m/sec while the vertical velocity forced to be zero. The lateral and longitudinal velocity of the pedestrian with respect to vehicle, shown in Figure 6.5a are estimated with an error of less than **0.5** m/sec and **0.02** m/sec^2 respectively. In addition, the estimated acceleration parameters that lead to the aforementioned velocities are shown in Figure 6.5b. As expected, the accelerations have a null mean value as the pedestrian is crossing at a constant velocity.

6.1.4 Experiment 2 - SDV in Motion

In this experiment, the object, which in this case is a pedestrian, crosses from left to right in the image frame as shown in Figure 6.3b. Note that when the vehicle is moving forward, it undergoes longitudinal deceleration as well as changes in yaw angle (non-zero yaw rate). Thus, it is necessary to compensate this motion of the SDV to obtain the absolute position of the pedestrian. The pedestrian's estimated position at each given timestep is shown in Figure 6.6. As seen in the experimental results, our approach decouples the effect of lateral and longitudinal motion of the SDV's motion from the observed pedestrian motion in the SDV frame of reference achieving a maximum position tracking error of less than **50** cm. As before, the estimated trajectory of the pedestrian is visualized in the WGS84 world

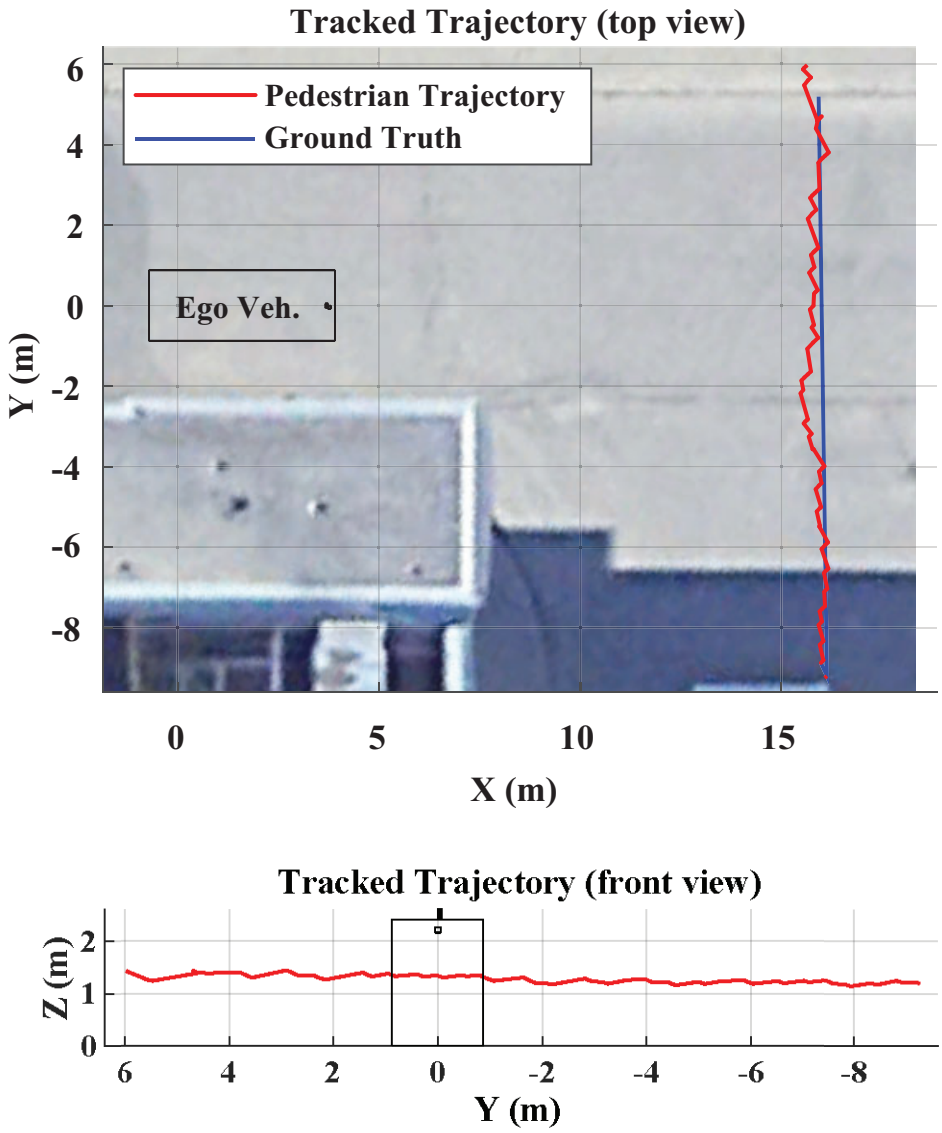
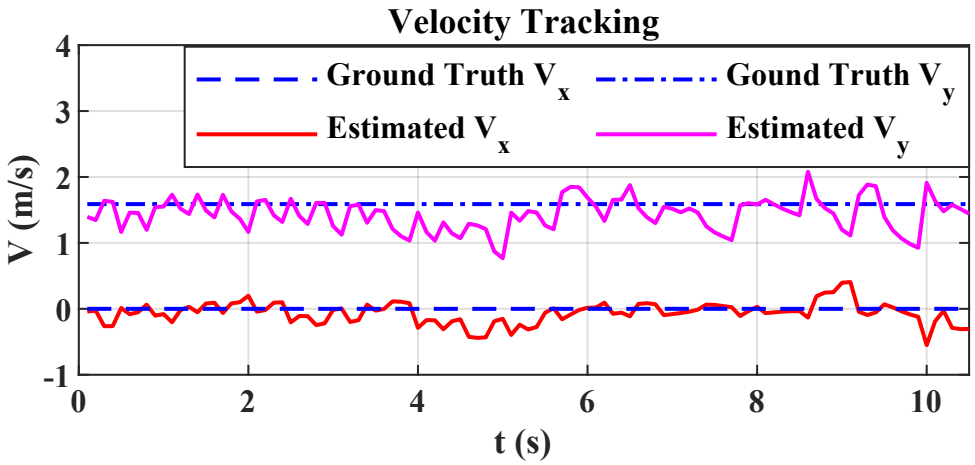
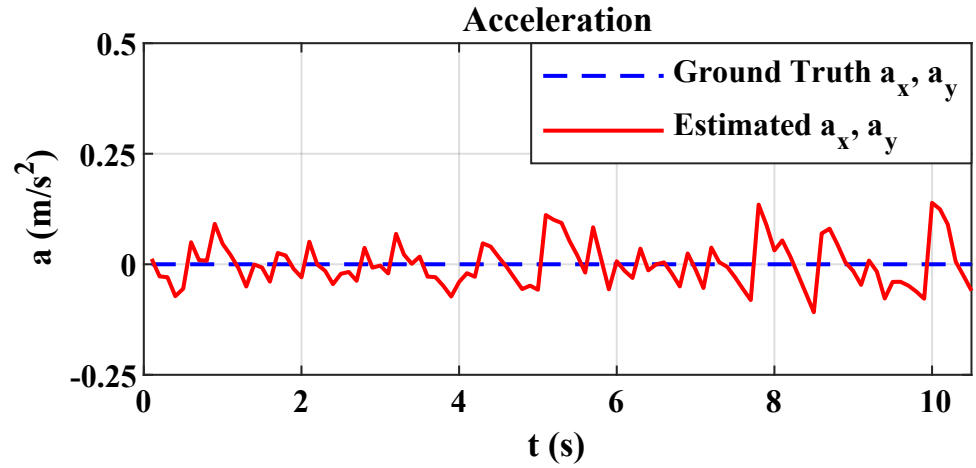


Figure 6.4: Position tracking for Case (1). The pedestrian starts roughly 15m, away on the right of the SDV, then follows a laterally straight trajectory.



(a) Velocity Tracking for Case (1)



(b) Acceleration Tracking for Case (1)

Figure 6.5: Velocity and acceleration estimation comparison for Case (1)

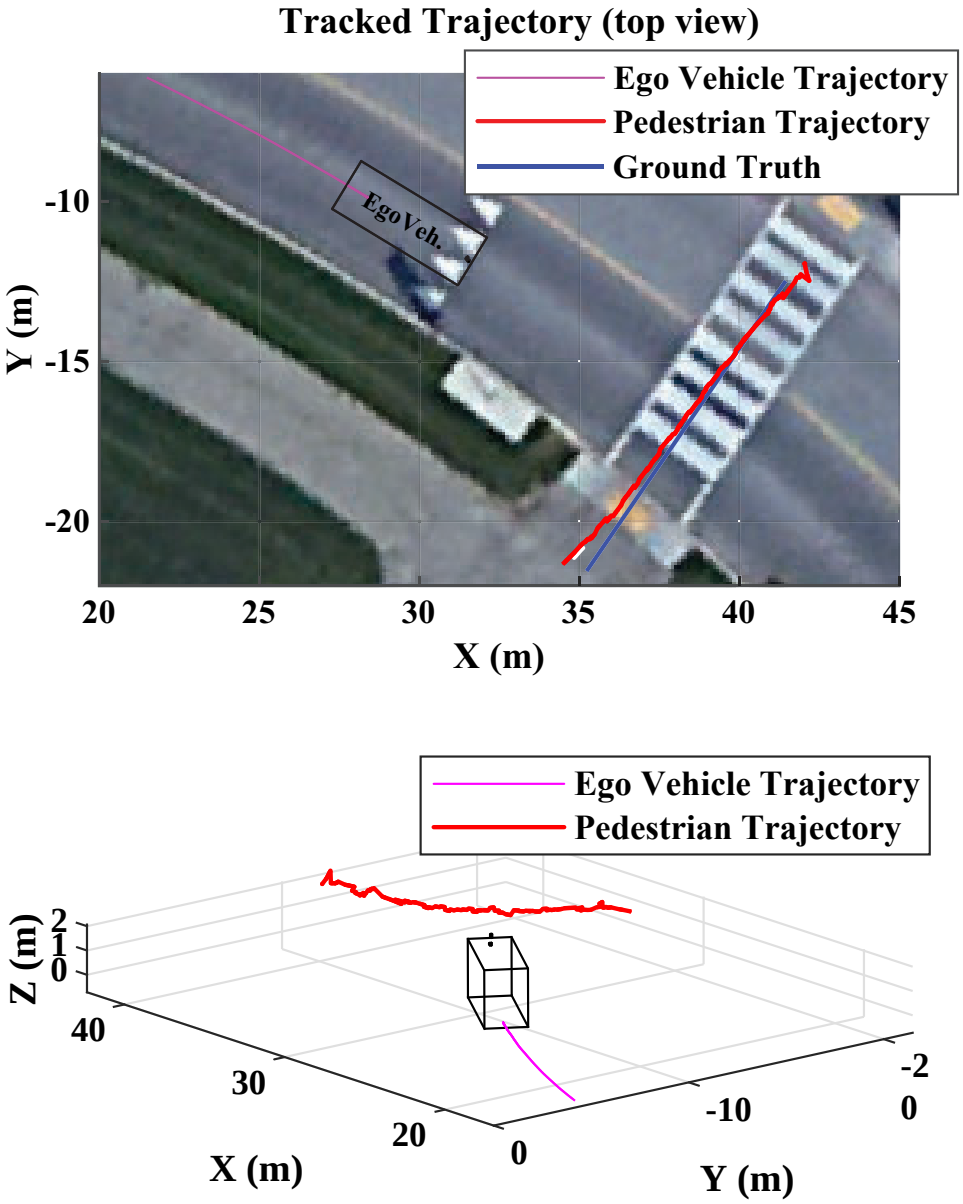
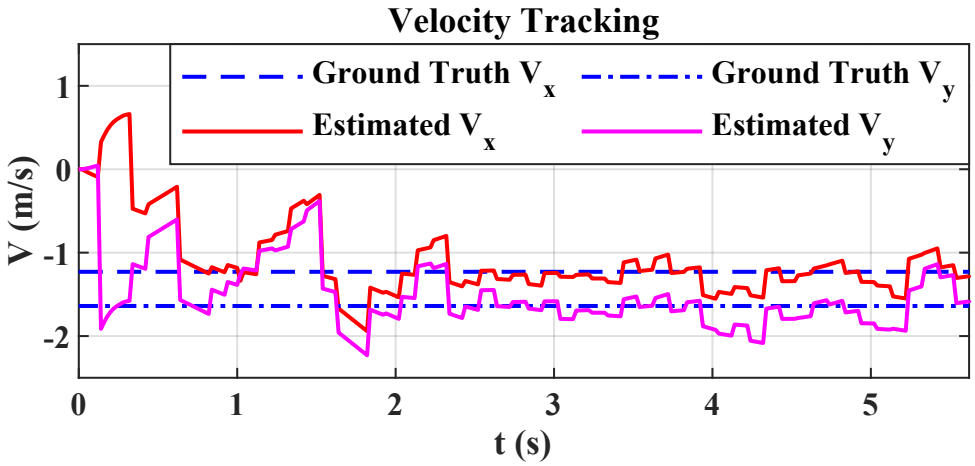
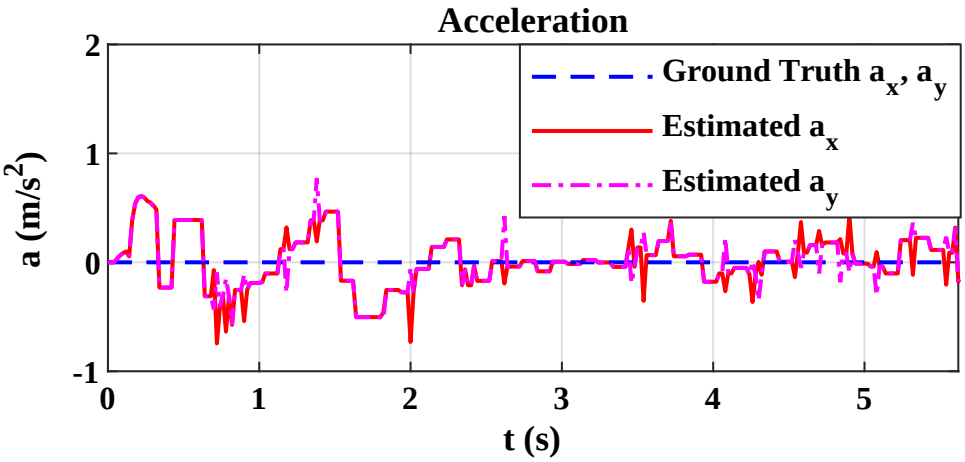


Figure 6.6: Position tracking for Case (2). The pedestrian starts roughly 25m, away on the left of the moving SDV, then follows a laterally straight trajectory.



(a) Velocity Tracking for Case (2)



(b) Acceleration Tracking for Case (2)

Figure 6.7: Velocity and acceleration estimation comparison for Case (2)

frame with an overlay on an open street map. Note that the estimated trajectory of the pedestrian closely follows the ground truth trajectory as shown in Figure 6.6. In addition, the estimated centroid height of the pedestrian is obtained as shown in Figure 6.6.

Furthermore, utilizing the same bounded inequality constraints for the states as in Case (1), the lateral and longitudinal velocity of the pedestrian with respect to vehicle, shown in Figure 6.7a, is estimated with an error of less than **0.5** m/sec and **0.004** m/sec^2 respectively, when it is settled after $t = 2$ sec. In addition, the estimated acceleration parameters producing the aforementioned velocities are shown in Figure 6.7b.

6.2 MPC-PF: Experimental Results

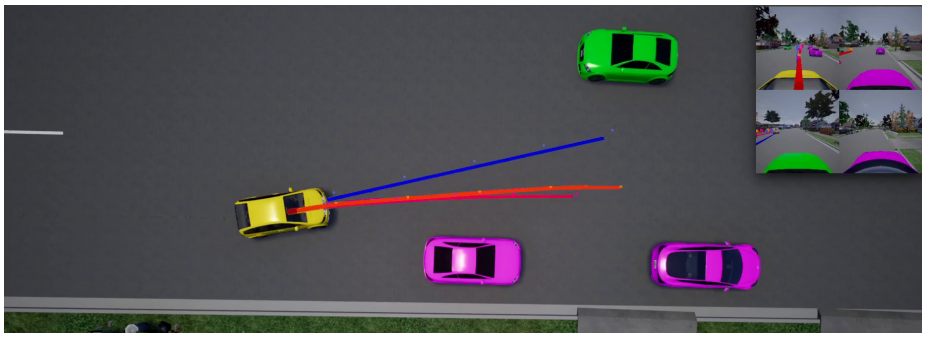
In this section, the effectiveness of MPC-PF on the Waymo Open Motion Dataset as well as on two unseen scenarios that autonomous vehicles encounter quite often in an urban setting is evaluated. Detailed quantitative and qualitative results for these scenarios are presented along with an ablation study (see supplementary **videos** at https://youtu.be/56sD-qsREBY and https://youtu.be/4DR7fqruLGo for further qualitative illustrations).

6.2.1 Evaluation Setup

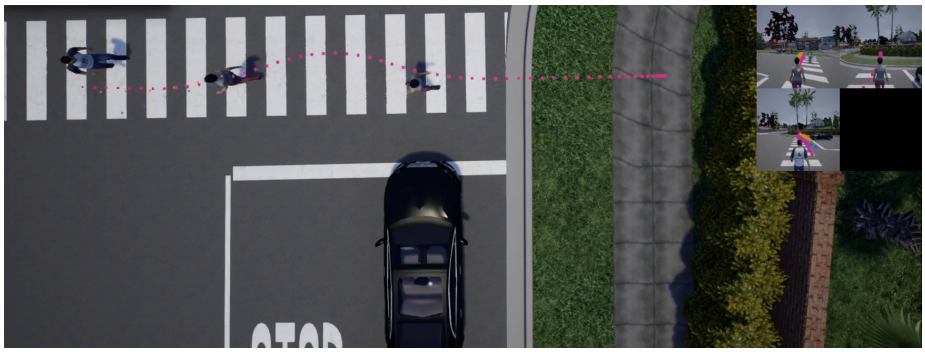
The proposed approach is evaluted on the Waymo Open Motion Dataset given that it contains over 100K sample segments, each with 20 seconds of object trajectory and map information data sampled at 10Hz. Any given sample consists of 1 second of prior information and 8 seconds of future trajectory ground truth for up to 8 agents of interest per sample. In addition, this dataset is a benchmark for trajectory prediction given that results from state-of-the-art approaches are readily available for comparison. In addition, to evaluate model generalization performance, state observations are generated for agents in two scenarios in simulation environment using the CARLA project in Unreal Engine 4.22 [83]. CARLA offers direct access to object position, velocity, and control which makes for a convenient simulation and test environment. In addition, the macad-gym project was used to package state variables and it offers an OpenAI gym-based interface which is convenient for testing purposes [84].

A sample illustration of the scenarios from both datasets is shown in Figure 6.8. For all tests performed, $h = 6, \Delta t = 0.3s, a = 1, b = 1.75, Q = I_h, r = 10^3, s = 50$ as they yield good empirical performance across scenarios. These parameters have an impact on the results obtained, however, they do not need to be tuned for each scenario. Specifically,

Scenario 1 - Oncoming vehicle "nudging" parked cars



Scenario 2 - Pedestrians at stop sign-based intersection



Scenario 3 and 4 - Samples from Waymo dataset

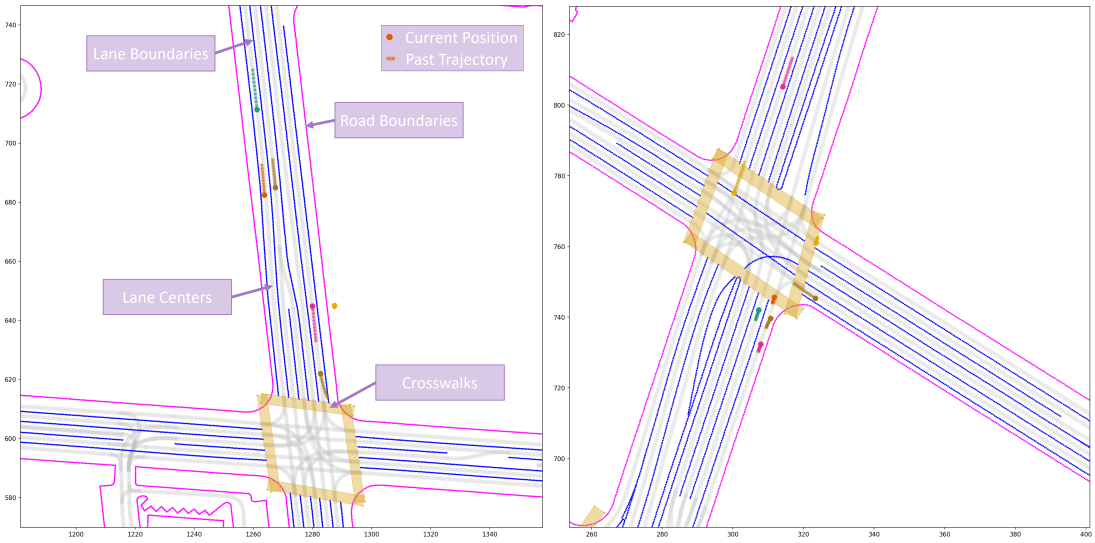


Figure 6.8: Illustration of evaluation scenarios consisting of predictions for both car and pedestrian object types.

this is since the potential fields are scaled to range between 0 and a fixed maximum value of 10. This ensures that the balance between each of the three terms in Equation (4.7) given that the cost incurred due to the potential field is on a fixed scale and that reference and heading deviation are based on the relative amount of deviation from nominal values based on physical constraints.

The algorithm, developed in python, was tested on a single machine with a Intel i7-9750 Processor and 16 GB RAM (comparatively low end). The computation of predicted trajectories is parallelized via multi-threading and the average processing time attributed to the computation for any given object is around 10ms.

Scenario 1: The SDV is in green and there are two parked vehicles on the road in magenta. In addition, a non-static vehicle in yellow driving along the same direction as the parked vehicles and opposite to that of the SDV performs a "nudge" maneuver to pass the parked vehicles. This scenario is particularly relevant in dense urban cities where narrow lanes require the nudging vehicle to enter the oncoming SDV vehicle lane and prompt appropriate decision to be taken. At the same time, if space allows and if the nudging vehicle can safely pass the parked vehicles while at the same time the SDV can also proceed, a stop decision for the SDV can be too conservative and block traffic making it difficult to navigate.

Scenario 2: Three pedestrians shown in green, magenta, and yellow are crossing at a stop sign controlled intersection and the SDV (not pictured in Figure 6.8) is on the opposite side of the intersection, however, a car is stopped near the pedestrians. In this scenario a sound prediction of the pedestrian trajectories is required to make a "go" decision. Furthermore, considering the interactions between the pedestrians can further aid in avoiding conservative path planning schemes which is elaborated on in the qualitative evaluation section to follow.

Scenario 3: Three vehicles are approaching an intersection while another vehicle travelling in the opposite direction is about to finish making a right turn and accelerating. In addition, a pedestrian is walking along the side walk. This scenario is a typical one that is often representative of an urban intersection. Predicting the motion of vehicles while under high lateral acceleration such as at turn is explored via this scenario.

Scenario 4: Several vehicles are near an intersection. There are four vehicles on the oncoming side with one vehicle intending to go straight. For the other three vehicles, there

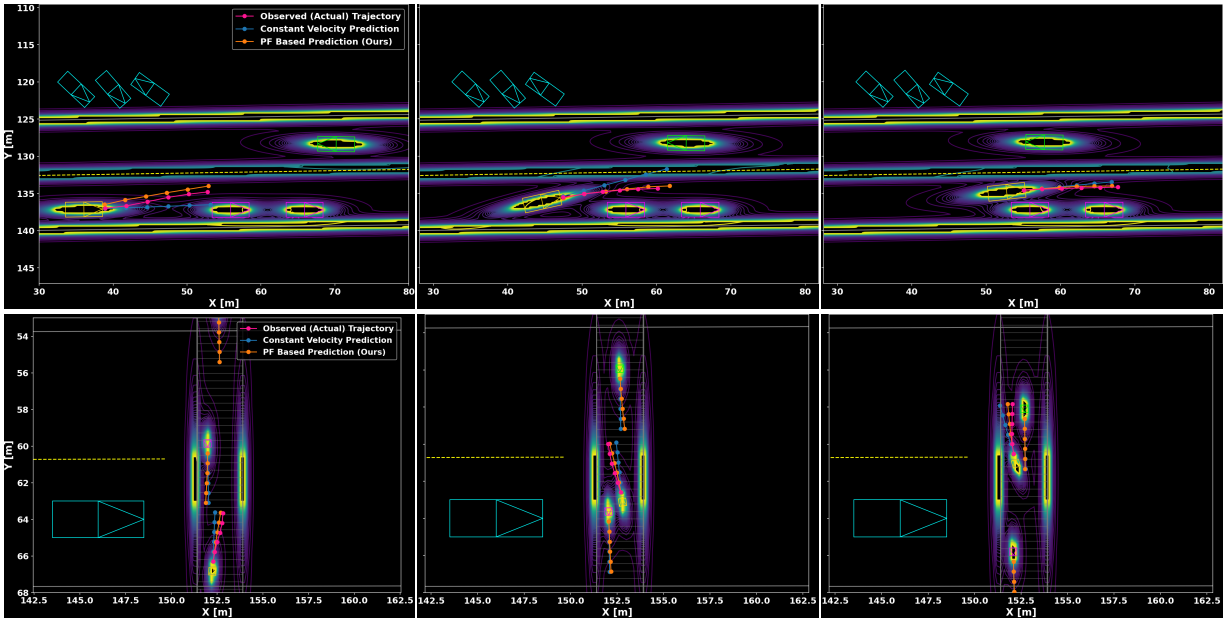


Figure 6.9: Qualitative illustration of model performance for Scenario 1 and 2 at three key timesteps across the scenario sequence.

exist two possible paths: going straight or making a right turn. This scenario is important as it evaluates the ability of the model to predict the most likely path of traversal based on a variety of factors.

6.2.2 Qualitative Evaluation

The qualitative results of MPC-PF on Scenarios 1-4 are showcased in Figures 6.9 and 6.10. The qualitative results for Scenarios 1 and 2 showcase three key snapshots across the sequence of timesteps.

In Figures 6.9 and 6.10, the magenta colored trajectory refers to the observed (actual) ground truth trajectory, which is the observation of the object positions directly from the environment. This trajectory acts as a baseline for comparing the trajectory prediction result with the proposed model, shown in orange, and the constant velocity trajectory, shown in blue. The error in the prediction can also be quantitatively measured using this baseline trajectory.

In addition, the potential fields for vehicles as well for road and lane boundaries for

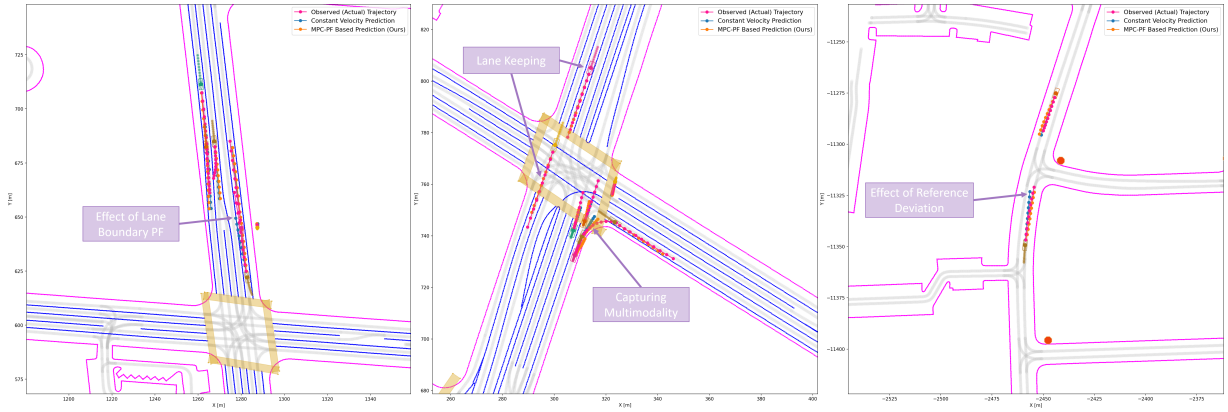


Figure 6.10: Qualitative illustration of model performance for Scenario 3 and 4 from Waymo Dataset.

Scenarios 1 and 2 are shown in Figure 6.9 while those for Scenarios 3 and 4 are shown in Figure 4.4. In Figure 4.4 green denotes lower costs and red denotes higher costs.

Scenario 1: When the nudging vehicle approaches the parked cars on the road, it incurs a high potential field cost due to the presence of the parked car nearby. In addition, there exists a potential field for the road boundaries shown in grey and a relatively low cost potential field for the center lane shown in yellow. Consequently, the predicted trajectory for the nudging vehicle deviates from the constant velocity-based prediction in blue and closely matches the actual trajectory observation. Furthermore, when the nudging vehicle's heading faces towards the SDV in green, the constant velocity trajectory is yet again seen to be inconsistent with ground truth trajectory and can cause over-conservative decisions made by the SDV. The proposed model, however, matches closely to the ground truth trajectory due to the presence of the low cost center lane potential field and that of the parked vehicles. Lastly, when the nudging vehicle is beside one of the parked cars, the predicted trajectory deviates from the center line trajectory of the lane.

Scenario 2: The predicted trajectories of all three pedestrians in the scenario demonstrate the applicability of the proposed approach to multi-object scenarios. When the pedestrian in yellow approaches the pedestrian in magenta and a collision is anticipated, due to the potential field cost embedded in the trajectory prediction of both the pedestrians, the predicted trajectory of both pedestrians deviate from the constant velocity trajectory. However, the predicted trajectory of the pedestrian in yellow does so more given that the

pedestrian in magenta is already at the edge of the crosswalk. Given that there exists a low cost potential field at the edge of the crosswalk, it is the pedestrian in yellow that is anticipated to give way to the pedestrian in magenta. Furthermore, when the pedestrian in yellow encounters the presence of the pedestrian in green, the predicted trajectory of the yellow pedestrian deviates from the constant velocity trajectory and matches the actual trajectory. Lastly, when the pedestrian in yellow has a heading leading out of the crosswalk, as seen by the constant velocity trajectory prediction, the presence of the low cost crosswalk potential field enables a prediction that matches closely with the actual trajectory. Having accurate trajectory predictions that incorporate interactions between pedestrians and cars as well the road map is important for the SDV path planning. For instance, in this scenario having accurate pedestrian trajectory predictions (informed by the potential field of the crosswalks and road boundaries) as well that of the car at the stop sign can help the SDV plan a path and execute a "go" decision while the green pedestrian is crossing in front of the car waiting at the stop sign and hence avoid over-conservative decisions.

Scenario 3: As the vehicle (in brown) completes making a right turn, it incurs a potential field due to the unalignment of the vehicle's heading to that of the lane boundaries. Thus, the predicted trajectory of the vehicle is closer to the reference path in order to obtain a balance between the potential field cost and reference deviation. Moreover, it is evident that the constant velocity prediction would imply that the vehicle would intend to merge directly into the next lane which is far from the ground truth. This shows the value that incorporation of the potential field has in trajectory prediction. Moreover, the predicted trajectories for the other vehicles and the pedestrian in the scenario also closely match the ground truth.

Scenario 4: Given that there are two possible paths (going straight or making a right turn) for the vehicles in orange, magenta, and brown, our approach correctly identifies the intention of all three vehicles. This is specifically due to the consideration of the current heading angle and speed of each of the vehicles as well as potential field due to the lane boundary on the right and the road boundary on the left. For the vehicles making a right turn, the trajectory prediction depicting going straight would have a high cost incurred due to abrupt heading deviation. Furthermore, for the vehicle in magenta, there is some cost incurred due to being close to the vehicle in brown. Similarly, for the vehicle in orange, a predicted trajectory depicting a right turn would cause a higher heading deviation cost given its position and speed. Similarly, for the vehicle in yellow, the predicted path depicts lane keeping rather than merging into the path onto the right given that there would be a

higher heading deviation cost in such a case. Note that if, due to an unexpected reason, the vehicle was to merge onto the right lane, the model would be able to account for that as soon as there is an indication of heading angle change via the measured states.

6.2.3 Evaluation Metrics

To quantitatively evaluate the performance of the approach, Average Displacement Error (ADE), Final Displacement Error (FDE), and Average Precision (AP) are selected as the primary evaluation metrics .

ADE refers to the Mean Square Error (MSE), that is mean of the L_2 norm, between the ground truth object trajectory compared to the predicted trajectory at each timestep and for each step of the prediction horizon for all objects. It is computed via Equation (6.1) below.

$$ADE = \frac{1}{nh} \sum_{i=1}^{n} \sum_{T=t+1}^{t+h} \|\hat{\boldsymbol{y}}_T - \boldsymbol{y}_T\|_2 \tag{6.1}$$

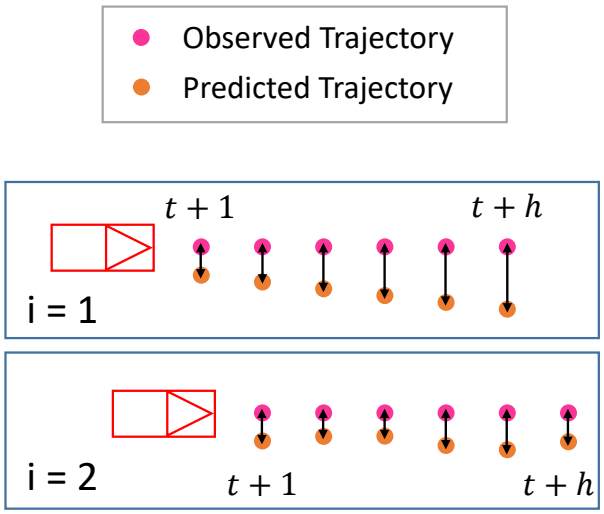
In addition, a visual illustration of the distances used for computing MSE for ADE is shown in Figure 6.11a.

FDE, on the other hand, is the MSE calculated between the position of the object in the final horizon timestep of the ground truth trajectory at a given timestep and that of the corresponding predicted position for all objects. It is computed via Equation (6.2) below.

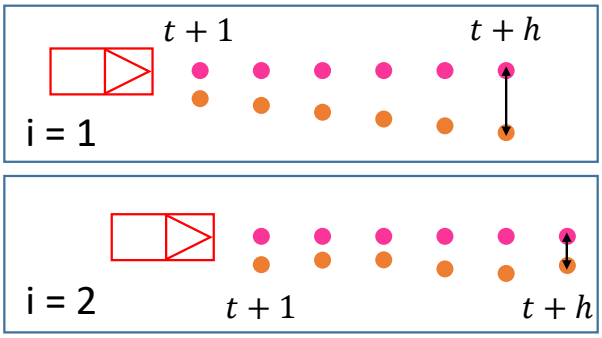
$$FDE = \frac{1}{n} \sum_{i=1}^{n} \|\hat{\boldsymbol{y}}_{t+h} - \boldsymbol{y}_{t+h}\|_2 \tag{6.2}$$

A visual illustration, similar to ADE, of the distances used for computing MSE for FDE is shown in Figure 6.11b.

To compute the AP, trajectory predictions are classified into true and false positives (TP and FP) using the notion of miss. A miss is defined as the state when the predicted trajectory for any object does not fall within a given lateral and longitudinal threshold of the ground truth trajectory at any given timestep. A visual illustration is shown in Figure 6.12.



(a) Illustration of the distances used for computing MSE in ADE.



(b) Illustration of the distances used for computing MSE in FDE.

Figure 6.11: Illustration of ADE and FDE metrics.

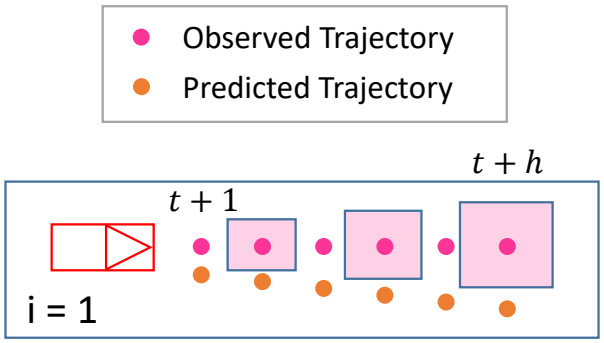


Figure 6.12: Illustration of the distances used for computing MSE in FDE.

Using this definition of a miss, any trajectory prediction that is classified as a miss is assigned a false positive and any that is not considered a miss is assigned a true positive. Alike object detection AP metrics, a single true positive is allowed for each prediction. True and False Positives (TP, FP) as well as False Negatives (FN) are stored and the AP metric is computed via the determining the area under the Precision-Recall curve for n instances of observations via Equations (6.3) and (6.4).

$$Precision = \frac{TP}{TP + FP}, \; Recall = \frac{TP}{TP + FN} \tag{6.3}$$

$$AP = \frac{1}{n} \sum_{k=0}^{k=n} P_k R_k \tag{6.4}$$

Moreover, to calculate the quantitative metrics, the timesteps and longitudinal and lateral thresholds shown in Table 6.2 are selected which match those used in the Waymo Open Motion Dataset [16]. The timestep evaluations for Scenario 1 and 2, however, are at 0.6, 1.2, and 1.8 seconds with base longitudinal thresholds of 0.5, 1, and 1.625m and lateral thresholds of 0.2, 0.225, and 0.25m respectively which in fact is less tolerant in terms of accepted error. These metrics are computed at each timestep in the prediction horizon to evaluate the performance of the proposed MPC-PF model.

The thresholds are also scaled based on the speed of the agent via a piecewise linear scaling function as shown in Equation (6.5) below.

Table 6.2: Lateral and longitudinal thresholds for AP calculation.

Timestep [s]	$Threshold_{lat}$ [m]	$Threshold_{long}$ [m]
T = 3	1	2
T = 5	1.8	3.6
T = 8	3	6

$$Scale(\boldsymbol{v}_i) = \begin{cases} 0.5 & if \quad \boldsymbol{v}_i < 1.4m/s \\ 0.5 + 0.5\alpha & if \quad 1.4m/s < \boldsymbol{v}_i < 11m/s \\ 1 & if \quad \boldsymbol{v}_i > 11m/s \end{cases} \tag{6.5}$$

$$\text{where } \alpha = (\boldsymbol{v}_i - 1.4)/(11 - 1.4)$$

Thus, the effective lateral and longitudinal thresholds are calculated as in Equations (6.6) and (6.7) below.

$$Threshold_{lat}(\boldsymbol{v}_i, T) = Scale(\boldsymbol{v}_i) \times Threshold_{lat}(T) \tag{6.6}$$
$$Threshold_{long}(\boldsymbol{v}_i, T) = Scale(\boldsymbol{v}_i) \times Threshold_{long}(T) \tag{6.7}$$

6.2.4 Quantitative Evaluation

The performance of the proposed model is reported on the Waymo Open Motion Dataset and the two scenarios discussed above. The evaluation for the Waymo Dataset scenarios is done per the dataset specification for metric calculations. A comparison of the performance of our approach compared to state-of-the-art approaches is presented in Table 6.3. For Scenario 1 and 2, the evaluation metrics are reported for trajectories observed 2, 4, and 6 timesteps in the future corresponding to 0.6 seconds, 1.2 seconds, and 1.8 seconds in Table 6.4. In particular, a comparison of the proposed method using ADE, FDE, and AP and an ablation study of the model is presented to justify generalization and the significance of each of the three terms in Equation (4.7).

Our approach is able to achieve close performance to the top state-of-the-art methods that achieved 1^{st}, 2^{nd}, and 3^{rd} in the 2021 and 2022 competitions. In particular, the proposed approach achieves better AP results compared to the 3^{rd} place winner for the

Table 6.3: Quantitative performance comparison of MPC-PF with state-of-the-art methods on Waymo Open Motion Dataset.

Method	ADE [m] ↓	FDE [m] ↓	AP ↑
MTR-A (1^{st} 2022) [60]	0.5640	1.1344	0.4492
golfer (2^{nd} 2022) [53]	0.5533	1.1608	0.4119
HBEns (3^{rd} 2022)	0.6431	1.3405	0.3700
DM (2022)	0.6777	1.3558	0.3710
Multipath++ (2021) [54]	0.5557	1.1577	0.4092
DenseTNT (1^{st} 2021) [69]	1.0387	1.5514	0.3281
ReCoAt (2^{nd} 2021) [55]	0.7703	1.6668	0.2711
SimpleCNNOnRaster (3^{rd} 2021) [56]	0.7400	1.4936	0.2136
AE-LSTM (2021)	2.3011	5.8579	0.0885
MPC-PF	1.0102	1.6252	0.3105

Table 6.4: Quantitative performance of MPC-PF on Scenario 1 and 2 along with model ablation.

Method	Scenario 1 - Oncoming vehicle "nudging" parked cars								
	ADE [m] ↓			FDE [m] ↓			AP ↑		
	0.6s	1.2s	1.8s	0.6s	1.2s	1.8s	0.6s	1.2s	1.8s
Constant Velocity	0.251	0.531	0.869	0.347	1.010	1.733	0.643	0.429	0.714
MPC-PF w/o proximity	0.760	0.843	0.906	0.835	0.950	1.390	0.786	0.571	0.813
MPC-PF	**0.221**	**0.442**	**0.668**	**0.319**	**0.745**	**1.260**	**0.857**	**0.643**	**0.861**
	Scenario 2 - Pedestrians at stop sign-based intersection								
Constant Velocity	0.036	0.094	0.169	0.056	0.189	0.364	0.750	0.721	0.692
MPC-PF w/o proximity	0.119	0.128	0.166	0.124	0.142	0.314	0.575	0.500	0.400
MPC-PF	**0.025**	**0.064**	**0.121**	**0.038**	**0.130**	**0.274**	**0.875**	**0.775**	**0.725**

2021 competition. Note that while most of these approaches use a purely data-driven approach to achieve the reported performance, our approach is driven by incorporating intuitive prior information into a structured model via a cost function. In addition, our approach is able to generalize well to other scenarios such as in Scenario 1 and 2 without the need to change parameters or remodel the cost function. The state-of-the-art approaches generally need to be retrained or may perform inadequately in these unseen scenarios as they are not guided by a structured model that is directly constructed based on intuitive behaviours.

The ADE and FDE metrics increase with increase in the prediction timestep as expected. Note that there are different longitudinal and lateral tolerance thresholds for calculation of AP and that the thresholds increase as the timestep increases. Due to this, the AP values are not expected to have a monotonous increase as timestep increases, especially since the change in the overall timestep is not drastically high. The full MPC-PF model outperforms the model variants with one or more terms excluded from that formulated in Equation (4.7) in all of the metrics. Specifically, the AP of the full MPC-PC model is significantly higher than both of the other models. It is important to note that having a reference trajectory either from a constant velocity-based trajectory estimate for pedestrians or through center lane information for cars is essential for obtaining an initial base case prediction. Specifically for pedestrians, there is no given center line to track along and hence the constant velocity-based trajectory estimate is used as a baseline on which realizations of deviation may occur due to presence of surrounding objects.

Significance of Potential Field - PF Cost: The Constant Velocity model is obtained when the heading deviation and PF cost terms are removed from Equation (4.7). It is important to note that the FDE of the Constant Velocity model at 1.8 and 1.2 seconds is the largest amongst all models and this is primarily due to the absence of potential field information for both surrounding objects and features from the road map. This is especially true during cases where the heading angle of the object for which the prediction is desired points away from the expected lane center or reference path such as for the vehicle in brown for Scenario 3 and in the first snapshot of Scenario 1 in Figure 6.10. This also occurs when the heading angle points away from a road map feature boundary such as in the second snapshot of Scenario 1 and second snapshot of Scenario 2. In contrast, MPC-PF without the proximity term outperforms the Constant Velocity model in terms of FDE due to inclusion of the potential field. In terms of ADE at 1.8 seconds, however, there is minimal difference. However, in terms of AP, MPC-PF outperforms the Constant Velocity model by 15% in Scenario 1 and MPC-PF without proximity by 32% in Scenario 2.

Significance of Proximity - Heading and Reference Deviation: MPC-PF without proximity outperforms the Constant Velocity model in terms of FDE for 1.8 and 1.2 seconds, however, the opposite is true for 0.6 seconds. This is precisely because of the lack of the proximity cost and it is also reflected in the ADE for 0.6 and 1.2 seconds. The Constant Velocity model performs better for short term prediction as it is entirely based on proximity - in fact it assumes fixed heading across the entire prediction horizon. Thus, only a "soft" proximity cost combined with the PF cost enables dynamic tradeoff between the two terms across the prediction horizon. Note that for the case of pedestrians, the proximity term is especially important given that their velocity is comparatively low and this is reflected by the lower AP scores of this model compared to the Constant Velocity model or the full MPC-PF model.

Informed by the significance of proximity (via both the heading and reference deviation terms) especially for short term prediction and that of the PF cost for long term, the full MPC-PF model performs the best and is able to dynamically tradeoff between the costs across the prediction horizon.

6.3 Associated Videos

A collection of additional **videos** that provide qualitative illustrations are available at https://www.youtube.com/playlist?list=PLQG3T3nkAnJ8qxJz8-jpqMEWN5JEh0Wt3.

6.4 Summary

In this chapter, comprehensive results for both the MHE and MPC-PF schemes were presented. The results from MHE show a maximum position estimate error of 50 cm and maximum velocity tracking error of 0.5 m/sec for both cases confirming the capability of the proposed approach in solving the challenges identified earlier. Furthermore, the results for MPC-PF on Scenario 1 and 2 showcase promising results in terms of ADE, FDE, and AP metrics.

More importantly, the results on Scenario 3 and 4 from the Waymo dataset showcase state-of-the-art results compared to the top three approaches on the 2022 and 2021 leaderboards. Moreover, it is important to mention the generalization capability of the approach compared to state-of-the-art approaches which generally need to be retrained on

unseen scenarios. Thus, both the MHE and MPC-PF models help accomplish the research objectives of this thesis.

Chapter 7

Conclusions and Future Work

The primary objective of this thesis project was development of a socially and spatially aware motion prediction framework for dynamic objects. In contrast to state-of-the-art methods, the proposed framework provides a structured yet adaptive approach to model intuitions on social interaction while remaining generalizable to several scenarios that an autonomous vehicle may encounter.

7.1 Conclusions

In this thesis, the first objective of 3D position and velocity estimation of objects was achieved via the design and implementation of the MHE scheme. Using a real-time LIDAR to image plane projection and fusion scheme, the depth association search space was reduced. Moreover, odometry information from GPS/IMU was used to localize the SDV and decouple the motion of the SDV from the perceived motion of objects enabling the estimation of the position and velocity of surrounding objects of interest in a global frame. Utilizing this scheme, the 3D position measurements for each object were obtained. Taking these intermittent or noisy measurements as inputs, the finite horizon MHE framework enabled online estimation of object position, velocity, and acceleration via a constrained optimization cost function. The approach enabled physical constrains to be imposed and dealt with intermittent loss in sensor measurements accomplishing the objectives aforementioned.

The experimental results for the MHE algorithm implemented on the WATonoBus platform confirm the applicability of the approach given that the position and velocity

are estimated within 50 cm and 0.5 m/sec error respectively. In addition, the results demonstrate its ability to handle intermittent sensor measurement losses, decouple effect of ego vehicle's motion from observed relative motion of objects, as well as co-estimate motion parameters through constrained optimization. Compared, to one-step lookback-based filters or estimators, the approach provides a stable estimation due to consideration of a larger temporal window or measurements. In addition, the approach enables explicit incorporation of physical constrains which data-driven approaches lack.

In addition, the second primary objective of trajectory prediction for objects was achieved through the design and implementation of MPC-PF. The proposed approach embeds surrounding object and road map information in the form of a potential field to model agent-agent and agent-space interactions. Unlike pure social force-based approaches, MPC-PF allows for a dynamic tradeoff between the constant velocity-based reference trajectory and the cost incurred due to the presence of surrounding objects or road features while maintaining proximity in object heading predictions.

In addition, the experimental results for the potential field-based MPC-PF trajectory prediction algorithm depicted various scenarios relevant to and useful in autonomous decision making in urban settings covering multiple objects. The efficacy of the proposed multi-object trajectory prediction method was highlighted both qualitatively and quantitatively achieving results at par with top state-of-the-art approaches on the Waymo Open Motion Dataset and also on other common urban driving scenarios depicting social interaction. Moreover, through an ablation study, the significance of the heuristics embedded in the formulation was also justified. The generated trajectory predictions incorporate position and velocity information of objects from the past to inform future predictions. Through the design of potential fields incorporating scene-based context such as lane, road, and crosswalk boundaries the approach is able to capture social interactions amongst objects as well as the scene.

Thus, the proposed MHE and MPC-PF framework addresses the aforementioned three challenges in position and velocity estimation along with two key challenges in trajectory prediction and provides a practical solution to estimate states and model social interactions.

As an extension, the DDPG agent presented in this thesis also provides a hybrid solution to trajectory prediction. The qualitative results presented for the urban crossing scenarios demonstrate that the potential field concept is applicable to data-driven approaches via its integration in the reward function.

7.2 Future Work

The research performed till date has umpteen avenues for further development and integration. Hence, some improvements that may form future work are considered.

Adaptive and Online Parameter Tuning For Estimation

First, the optimal system tuning parameters for MHE (affecting settling time) are not always attained since they currently are not estimated online, but are rather constant. Secondly, in an urban crowded crossing, the depth association approach may perform poorly as the centroid depth measurements maybe inaccurate in cases where occlusion occurs. Hence, future work should include implementing an online tuning/weight estimator that maintains optimal system parameters in multiple driving scenarios. In addition, a robust tracking mechanism for scenarios where multiple occlusions may occur can be integrated with the approach.

Robustness and Scalability Evaluation for Edge Cases

The robustness of the proposed approach and optimization techniques to different starting positions, velocities, and heading angles must be further verified by running experiments in real-time for various edge cases – that is with varying number of objects and different types of initial positions and velocities. In particular, the following studies are to be evaluated:

- Robustness of the optimization methods to various initial positions, velocities, and heading observations that include sensor noise and uncertainties
- Robustness of the trajectory prediction algorithm to acceleration estimation error from MHE due to object detection or system parameter changes during discretization
- Robustness to multiple types of object social interactions such as on motorways

Hybrid, Probabilistic, and Multimodal Prediction

As aforementioned, in the insights derived from related works, relying on a fixed set of models may generalize well to a variety of cases, but have an inherent bias in the trajectory predictions. On the other hand, relying purely on observations from data through CNNs or RNNs may not generalize well to a variety of cases. It is anticipated that the potential field-based prediction approach would benefit from added flexibility and multi-modality in the developed heuristics through learned observations and hence points towards a fusion between this model-based approach with a data-driven method. Thus, a hybrid algorithm

that is loosely governed by a fixed set of rules imposed by a model that allows the flexibility to predict from learned observations can potentially offer a balanced solution to trajectory prediction. In particular, the RL-PF framework has a strong potential as demonstrated in Chapter 5 and can be further extended for multi-modality and tested for scalability.

Such an approach can also yield multimodal trajectories with an estimate of probability due to the data-driven nature of the method which can aid the decision making module downstream in being aware of several contingency plans.

Co-Design With Up and Downstream Modules

Given that autonomous driving systems comprise of various modules, evaluation of the effect of the proposed trajectory prediction approach on upstream perception tasks and downstream decision-making and path planning modules is essential. In addition, propagation of metrics such as level of uncertainty of the proposed approach can benefit and enhance downstream performance and can in turn be used as a feedback mechanism to enhance prediction performance.

Letter of Copyright Permission

IEEE, as the publisher of the three aforementioned manuscripts fully or partly adopted in Chapter 3, 4, 6, allows the reuse of published papers in theses without formal permissions. The waivers of copyright from IEEE are achieved by display of the following statement:

Policy Regarding Thesis/Dissertation Reuse, from IEEE Copyright Clearance Centre

"The IEEE does not require individuals working on a thesis to obtain a formal reuse license; however, you may print out this statement to be used as a permission grant".

References

[1] Neel P. Bhatt, Amir Khajepour, and Ehsan Hashemi. MPC-PF: Socially and Spatially Aware Object Trajectory Prediction for Autonomous Driving Systems Using Potential Fields. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 2023.

[2] Neel P. Bhatt, Amir Khajepour, and Ehsan Hashemi. MPC-PF: Social Interaction Aware Trajectory Prediction of Dynamic Objects for Autonomous Driving Using Potential Fields. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9837–9844, 2022.

[3] Ehsan Mohammadbagher, Neel P. Bhatt, Ehsan Hashemi, Baris Fidan, and Amir Khajepour. Real-time Pedestrian Localization and State Estimation Using Moving Horizon Estimation. In *23rd Intelligent Transportation Systems Conference (ITSC)*, 2020.

[4] Alex Teichman and Sebastian Thrun. Practical object recognition in autonomous driving and beyond. In *Advanced Robotics and its Social Impacts*, pages 35–38, 2011.

[5] Jerome Lutin, Alain Kornhauser, and Eva Lerner-Lam. The revolutionary development of self-driving vehicles and implications for the transportation engineering profession. *Ite Journal*, 83:28–32, 07 2013.

[6] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: a survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[7] Mahir Gulzar, Yar Muhammad, and Naveed Muhammad. A survey on motion prediction of pedestrians and vehicles for autonomous driving. *IEEE Access*, 9:137957–137969, 2021.

[8] Kai Yu, Liqun Peng, Xue Ding, Fan Zhang, and Minrui Chen. Prediction of instantaneous driving safety in emergency scenarios based on connected vehicle basic safety messages. *Journal of Intelligent and Connected Vehicles*, ahead-of-print, 11 2019.

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

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[11] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168. IEEE, 2011.

[12] Anand Gopalakrishnan, Ankur Mali, Dan Kifer, Lee Giles, and Alexander G. Ororbia. A neural temporal model for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[13] Tim Lebailly, Sena Kiciroglu, Mathieu Salzmann, Pascal Fua, and Wei Wang. Motion prediction using temporal inception module. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020.

[14] Benjamin Völz, Holger Mielenz, Igor Gilitschenski, Roland Siegwart, and Juan Nieto. Inferring pedestrian motions at urban crosswalks. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):544–555, 2019.

[15] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[16] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[17] Ronny Hug, Stefan Becker, Wolfgang Hübner, and Michael Arens. Particle-based pedestrian path prediction using lstm-mdl models. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2684–2691. IEEE, 2018.

[18] Yanliang Zhu, Deheng Qian, Dongchun Ren, and Huaxia Xia. Starnet: Pedestrian trajectory prediction using deep neural network in star topology. *arXiv preprint arXiv:1906.01797*, 2019.

[19] Hao Xue, Du Q Huynh, and Mark Reynolds. Pedestrian tracking and stereo matching of tracklets for autonomous vehicles. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–5. IEEE, 2019.

[20] Nicolas Schneider and Dariu M Gavrila. Pedestrian path prediction with recursive bayesian filters: A comparative study. In *German Conference on Pattern Recognition*, pages 174–183. Springer, 2013.

[21] Vasileios Lefkopoulos, Marcel Menner, Alexander Domahidi, and Melanie N. Zeilinger. Interaction-aware motion prediction for autonomous driving: A multiple model kalman filtering scheme. *IEEE Robotics and Automation Letters*, 6(1):80–87, 2021.

[22] Sarah Ferguson, Brandon Luders, Robert C Grande, and Jonathan P How. Real-time predictive modeling and robust avoidance of pedestrians with uncertain, changing intentions. In *Algorithmic Foundations of Robotics XI*, pages 161–177. Springer, 2015.

[23] Christoph G Keller and Dariu M Gavrila. Will the pedestrian cross? a study on pedestrian path prediction. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):494–506, 2013.

[24] Peter Khomchuk, Inna Stainvas, and Igal Bilik. Pedestrian motion direction estimation using simulated automotive mimo radar. *IEEE Transactions on Aerospace and Electronic Systems*, 52(3):1132–1145, 2016.

[25] Paulo Vinicius Koerich Borges, Robert Zlot, and Ashley Tews. Integrating off-board cameras and vehicle on-board localization for pedestrian safety. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):720–730, 2013.

[26] Peixin Xue, Jianyi Liu, Shitao Chen, Zhuoli Zhou, Yongbo Huo, and Nanning Zheng. Crossing-road pedestrian trajectory prediction via encoder-decoder lstm. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2027–2033. IEEE, 2019.

[27] Naveed Muhammad and Björn Åstrand. Intention estimation using set of reference trajectories as behaviour model. *Sensors*, 18(12):4423, 2018.

[28] Ajay Jain, Sergio Casas, Renjie Liao, Yuwen Xiong, Song Feng, Sean Segal, and Raquel Urtasun. Discrete residual flow for probabilistic pedestrian behavior prediction. *arXiv preprint arXiv:1910.08041*, 2019.

[29] Lamberto Ballan, Francesco Castaldo, Alexandre Alahi, Francesco Palmieri, and Silvio Savarese. Knowledge Transfer for Scene-specific Motion Prediction. *CoRR*, abs/1603.0, 2016.

[30] Pasquale Coscia, Francesco Castaldo, Franscesco A N Palmieri, Lamberto Ballan, Alexandre Alahi, and Silvio Savarese. Point-based path prediction from polar histograms. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1961–1967, 2016.

[31] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity Forecasting. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 201–214, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[32] Bolei Zhou, Xiaogang Wang, and Xiaoou Tang. Random field topic model for semantic region analysis in crowded scenes from tracklets. In *CVPR 2011*, pages 3441–3448, 2011.

[33] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 7(3):652–674, 2022.

[34] Massimo Bertozzi, Alberto Broggi, Alessandra Fascioli, A Tibaldi, Roland Chapuis, and Freédéric Chausse. Pedestrian localization and tracking system with Kalman filtering. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 584–589, 2004.

[35] Andreas Møgelmose, Mohan M Trivedi, and Thomas B. Moeslund. Trajectory analysis and prediction for improved pedestrian safety: Integrated framework and evaluations. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 330–335, 2015.

[36] Joko Hariyono, Ajmal Shahbaz, and Kang-Hyun Jo. Estimation of walking direction for pedestrian path prediction from moving vehicle. In *2015 IEEE/SICE International Symposium on System Integration (SII)*, pages 750–753, 2015.

[37] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, may 1995.

[38] Laura Leal-Taixé, Gerard Pons-Moll, and Bodo Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 120–127, 2011.

[39] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 452–465, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[40] Narendra Ahuja and Jen-Hui Chuang. Shape representation using a generalized potential field model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):169–176, 1997.

[41] Jing Ren, Kenneth A. McIsaac, Rajni V. Patel, and Terry M. Peters. A potential field model using generalized sigmoid functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):477–484, 2007.

[42] Shuzhi Sam Ge and Youjing Cui. New potential functions for mobile robot path planning. *IEEE Transactions on Robotics and Automation*, 16(5):615–620, 2000.

[43] Jie Ji, Amir Khajepour, Wael William Melek, and Yanjun Huang. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. *IEEE Transactions on Vehicular Technology*, 66(2):952–964, 2017.

[44] Hong Shen and Ping Li. Unmanned aerial vehicle (uav) path planning based on improved pre-planning artificial potential field method. In *2020 Chinese Control And Decision Conference (CCDC)*, pages 2727–2732, 2020.

[45] Yalong Zhang, Zhenghua Liu, and Le Chang. A new adaptive artificial potential field and rolling window method for mobile robot path planning. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 7144–7148, 2017.

[46] Wang Di, Li Caihong, Guo Na, Song Yong, Gao Tengteng, and Liu Guoming. Local path planning of mobile robot based on artificial potential field. In *2020 39th Chinese Control Conference (CCC)*, pages 3677–3682, 2020.

[47] Ning Zhang, Yong Zhang, Chao Ma, and Bin Wang. Path planning of six-dof serial robots based on improved artificial potential field method. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 617–621, 2017.

[48] Li Zhou and Wei Li. Adaptive artificial potential field approach for obstacle avoidance path planning. In *2014 Seventh International Symposium on Computational Intelligence and Design*, volume 2, pages 429–432, 2014.

[49] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. TraPHic: Trajectory Prediction in Dense and Heterogeneous Traffic Using Weighted Interactions. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2019.

[50] Shengzhe Dai, Zhiheng Li, Li Li, Nanning Zheng, and Shuofeng Wang. A flexible and explainable vehicle motion prediction and inference framework combining semi-supervised aog and st-lstm. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):840–860, 2022.

[51] Sun Lihan, Yang Baoqing, and Ma Jie. A trajectory prediction algorithm for hfvs based on lstm. In *2021 40th Chinese Control Conference (CCC)*, pages 7927–7931, 2021.

[52] Raluca Didona Brehar, Mircea Paul Muresan, Tiberiu Mariţa, Cristian-Cosmin Vancea, Mihai Negru, and Sergiu Nedevschi. Pedestrian street-cross action recognition in monocular far infrared sequences. *IEEE Access*, 9:74302–74324, 2021.

[53] Xiaocheng Tang, Soheil Sadeghi Eshkevari, Haoyu Chen, Weidan Wu, Wei Qian, and Xiaoming Wang. Golfer: Trajectory prediction with masked goal conditioning mnm network, 2022.

[54] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S. Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, and Benjamin Sapp. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction, 2021.

[55] Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Recoat: A deep learning-based framework for multi-modal motion prediction in autonomous driving application, 2022.

[56] Stepan Konev, Kirill Brodt, and Artsiom Sanakoyeu. Motioncnn: A strong baseline for motion prediction in autonomous driving, 2022.

[57] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020.

[58] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning*, pages 895–904. PMLR, 2021.

[59] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020.

[60] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Mtr-a: 1st place solution for 2022 waymo open dataset challenge – motion prediction, 2022.

[61] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *ECCV*, 2020.

[62] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning, 2015.

[63] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P. How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350, 2017.

[64] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3949–3956, 2019.

[65] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *Int. J. Rob. Res.*, 36(10):1073–1087, sep 2017.

[66] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Neighbourhood context embeddings in deep inverse reinforcement learning for predicting pedestrian motion over long time horizons. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1179–1187, 2019.

[67] Eike Rehder, Florian Wirth, Martin Lauer, and Christoph Stiller. Pedestrian prediction by planning using deep neural networks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5903–5908, 2018.

[68] Brian D. Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J. Andrew Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936, 2009.

[69] Junru Gu, Qiao Sun, and Hang Zhao. Densetnt: Waymo open dataset motion prediction challenge 1st place solution, 2021.

[70] Peter Kühl, Moritz Diehl, Tom Kraus, Johannes P Schlöder, and Hans Georg Bock. A real-time algorithm for moving horizon state and parameter estimation. *Computers & chemical engineering*, 35(1):71–83, 2011.

[71] Dieter Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988.

[72] Shih-Ping Han. A globally convergent method for nonlinear programming. *Journal of optimization theory and applications*, 22(3):297–309, 1977.

[73] Michael JD Powell. The convergence of variable metric methods for nonlinearly constrained optimization calculations. In *Nonlinear programming 3*, pages 27–63. Elsevier, 1978.

[74] Michael JD Powell. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical analysis*, pages 144–157. Springer, 1978.

[75] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[76] Charles L Lawson and Richard J Hanson. *Solving least squares problems*. SIAM, 1995.

[77] Katherine M Mullen and Ivo HM van Stokkum. The lawson-hanson algorithm for non-negative least squares (nnls). *R Packag. version 1.4*, 2012.

[78] Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[79] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[80] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[81] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[82] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[83] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[84] Praveen Palanisamy. Multi-Agent Connected Autonomous Driving using Deep Reinforcement Learning, 2019.

[85] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[86] Ali Salimzadeh, Neel P. Bhatt, and Ehsan Hashemi. Augmented visual localization using a monocular camera for autonomous mobile robots. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 1124–1131, 2022.

[87] Daniel Flögel, Neel P. Bhatt, and Ehsan Hashemi. Infrastructure-aided localization and state estimation for autonomous mobile robots. *Robotics*, 11(4), 2022.

[88] Bruno H. Groenner Barbosa, Neel P. Bhatt, Amir Khajepour, and Ehsan Hashemi. Soft constrained autonomous vehicle navigation using gaussian processes and instance segmentation, 2021.

[89] Bogdan Oancea, Tudorel Andrei, and Raluca Mariana Dragoescu. Gpgpu computing, 2014.

[90] David Kirk et al. Nvidia cuda software and gpu parallel computing architecture. In *ISMM*, volume 7, pages 103–104, 2007.

[91] Marta Mrak. Ai gets creative, 2019.

[92] NVIDIA Corporation. Cublas library user guide, 2013.

[93] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning, 2014.

[94] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[95] Gary Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000.

[96] Jeremy Elson, John R. Douceur, Jon Howell, and Jared Saul. Asirra: a captcha that exploits interest-aligned manual image categorization. In *CCS '07*, 2007.

[97] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[98] Sergeant Wizard. Basic mnist example. https://github.com/pytorch/examples/blob/master/mnist/main.py, 2019.

[99] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S Vetter. Nvidia tensor core programmability, performance & precision. In *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pages 522–531. IEEE, 2018.

APPENDICES

Appendix A

Efficient Implementation of 2D Detectors Using CUDA and cuDNN on Tensor Cores

A.1 Introduction

The use of Graphics Processing Units (GPU) for General Purpose (GP) computing, in addition to the originally intended use of GPUs specifically for computer graphics, has increasingly been adopted in recent years. This spans various application domains, however, high performance scientific computing (HPC) applications have seen major benefits such as in training, Computer Aided Design (CAD), and Computational Fluid Dynamics (CFD) [89]. GPUs have been traditionally developed with support for execution of thousands of active threads, however, GP-GPU computing requires a domain specific programming model that can efficiently handle and express (in terms on machine level instructions) parallelism when supplied with high level user code.

Compute Unified Device Architecture (CUDA) is one such platform developed by NVIDIA Corporation specifically for their proprietary line of GPUs consisting of a collection of libraries that implement a programming model for parallel scheduling and execution of programs for GP-GPU computing.

Generally, a GPU consists of two main modules: (1) Global Memory (often referred to as GPU VRAM) and (2) Streaming Multiprocessors(SMs). When a user executes a program or an application that invokes the utilization of the CUDA library for a task

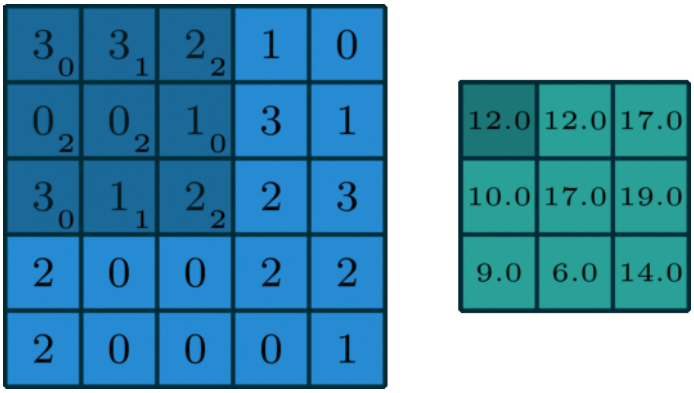


Figure A.1: This gif shows the convolution operation involving a 5x5 image (light blue), 3x3 kernel (dark blue), and a resulting 3x3 feature map (green). The operation involves element-wise multiplication of the 3x3 kernel with a 3x3 section of the image followed by sliding of the kernel in horizontal and vertical directions (**please view pdf in Adobe Acrobat Reader for the gif to play**).

that is to be executed on the GPU, the library, runs parallel portions of the program as a *kernel* on the GPU. A kernel dynamically allocates memory on the GPU (accessible by both the CPU and GPU) and manages the execution of the processes and threads based on resource availability and demand [90]. Modern GPUs consist of SMs equipped with multiple smaller processing units known as **CUDA cores** on which threads are executed. These cores are analogous to the cores present in the CPU, however, CUDA implements the parallel execution of these threads in a unique manner. A CUDA kernel is executed on an array of threads that are grouped into *blocks*. Within each block, threads cooperate through synchronization and can share memory. Furthermore, these blocks are then grouped into sequential *grids* and sent to the SMs for execution. The hardware architecture of the GPUs is designed to receive these grids of thread blocks and perform parallel computations.

Of particular interest in this thesis is the application of CUDA and other tools alike to develop and test DNNs. The nature of computations involved in training and testing of DNNs naturally lend themselves to be parallellized. Using efficient implementations can provide increased scope for training parameter search space given faster training performance. In recent years, CNNs have increasingly proven to be successful for image related reasoning in computer vision. CNNs involve a sequence of element-wise matrix multiplications between a kernel (usually smaller in dimension compared to the width and height of the image) and the image followed by sliding of the kernel over the image as shown in Figure A.1 [91]. Since, the element-wise multiplications per position of the kernel as well as

the multiplications arising from sliding the kernel (referred to as stride) are independent of each other they can be computed in parallel. While basic linear algebra operations can be parallelized using NVIDIA's CUDA Basic Linear Algebra Subprograms (cuBLAS) library, these specialized deep learning operations require paralellization of advanced routines [92].

CUDA Deep Neural Network (cuDNN) library provides an efficient implementation of frequently used deep learning routines. These include highly optimized forward and backward passes for routines such as convolution, max pooling, and activation functions [93]. This eliminates the need for the user to write these commonly used deep learning routines manually.

Since the introduction of cuDNN in 2014, support for these primitives was limited to operation with single (FP32) and double point precision (FP64) tensors. However, with the advent of the new *Turing* GPU architecture in 2018, efficient half precision (FP16) computations on specialized CUDA cores called **tensor cores** was made possible [99]. Tensor cores are specialized CUDA cores that are particularly used for efficient matrix multiplications. Specifically, tensors can compute a matrix multipy-accumulate operation which is a matrix multiplication of two 4x4 matrices plus a mixed precision matrix addition operation in one GPU clock as shown in Figure A.2. This amounts to 7 FP16 operations per row-column pair plus the scalar addition (arising from the matrix addition operation) totalling to 64 mixed precision operations per clock. This results in about **8x** faster computation time compared to that of CUDA cores with the trade-off of being capable to only compute this specific type of operation [99]. Nonetheless, since this occurs often in DNN workloads, it is a desired feature. Given availability of these two primary libraries, many computer vision and deep learning frameworks utilize these in order to perform efficient computations.

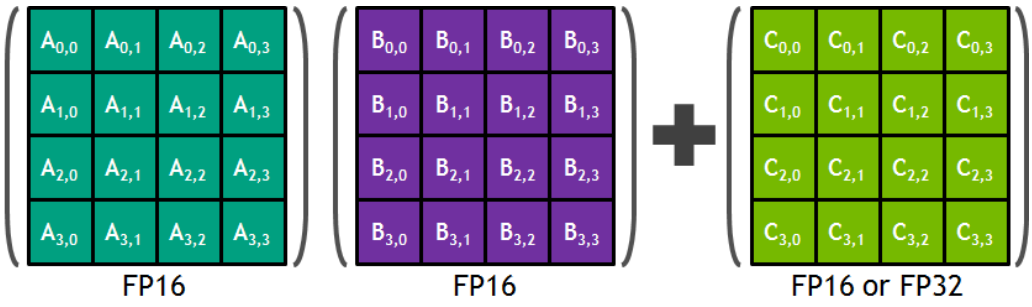


Figure A.2: Turing tensor cores are specialized CUDA cores that can perform half precision matrix multiply-accumulate operations in parallel - total of 64 operations per GPU clock compared to 8 operations on generic CUDA cores.

These include DNN development frameworks like PyTorch and image processing and I/O tools such as OpenCV [94][95]. The **(i)** CUDA and **(ii)** cuDNN (both with and without tensor cores) libraries are implemented and their performance (in terms of execution time) is empirically evaluated while running the YOLOv4 darknet network on the Microsoft COCO multi-class object detection dataset [96] [97].

A.2 Implementation and Runtime Comparison

The evaluation of the tools is conducted on the Microsoft COCO Multi-Class Object Detection Dataset for computation on these variants: (1) CPU, (2) GPU-CUDA, (3) GPU-CUDA+cuDNN, and (4) GPU-CUDA+cuDNN-FP16 (code written to utilize tensor cores). The hardware configuration used for the evaluation consist of: NVIDIA RTX2060 GPU, Intel i7-9750 CPU, and 16 GB RAM.

A.2.1 Microsoft COCO Multi-Class Object Detection Dataset

The COCO dataset consists of over 300 classes of common objects such as cars, pedestrians, and bicyclists. The two primarily tasks for the dataset are to perform multi-class classification and bounding box regression (fitting the best bounding box around the detected object).

The state-of-the-art object detection DNN for this task is the YOLOv4 model proposed in [85] as it offers a good trade-off between real-time performance and accuracy. Note that while *training* runtime performance was gauged in the previous dataset, here, the test runtime performance (measured by frames processed per second (FPS)) is to be explored. Thus, pre-trained weights and model configuration can be used directly for testing, only leaving conversion of the model and the input data to GPU and enabling cuDNN and tensor core acceleration. The re-implementation of the YOLOv4 DNN as outlined [85] is used.

To run the code on the GPU using CUDA, the tensor variables must be: (1) allocated using the **cudaMallocManaged** function, (2) the kernel *launch configuration* be set using: **kernelFunction** <<<**numThreadblocks, numThreadsperBlock**>>> and (3) free the memory using **cudaFree()** - variant (1).

In addition, for the cuDNN variant (2), setting the math mode to CUDNN_DEFAULT_MATH via the cudnnMathType_t enumerator for all tensors prompts the library to run in FP32

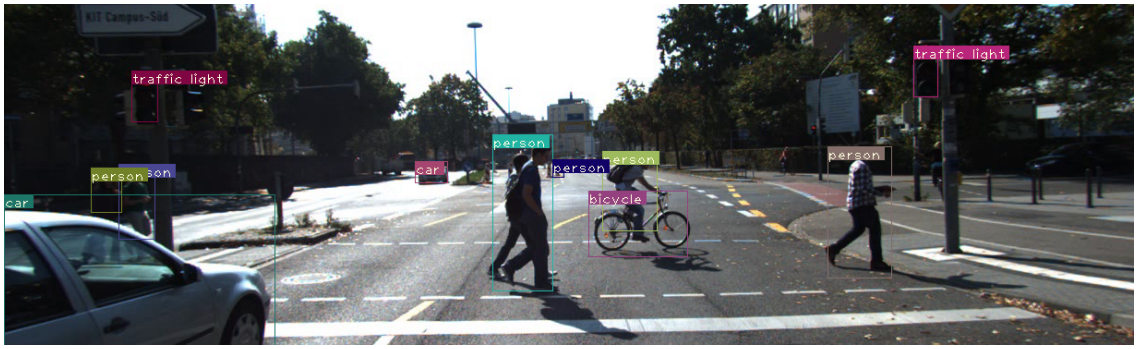


Figure A.3: Classified output detections and bounding boxes from Yolo4.

precision and hence will not use tensor cores. Furthermore, to enable tensor core (FP16) operations, the CUDNN_TENSOR_OP_MATH enumerator can be set to achieve variant (3). For both these variants, the **cudnnConvolutionForward()** and **cudnnConvolutionBackwardData()** functions can be used for the convolution operations in the network. Similar functions for max pooling and fully connected layers are available. Having these four variants, the runtime performance is evaluated. A sample image with output detections is depicted in Figure A.3 and performance results shown in Figure A.4.

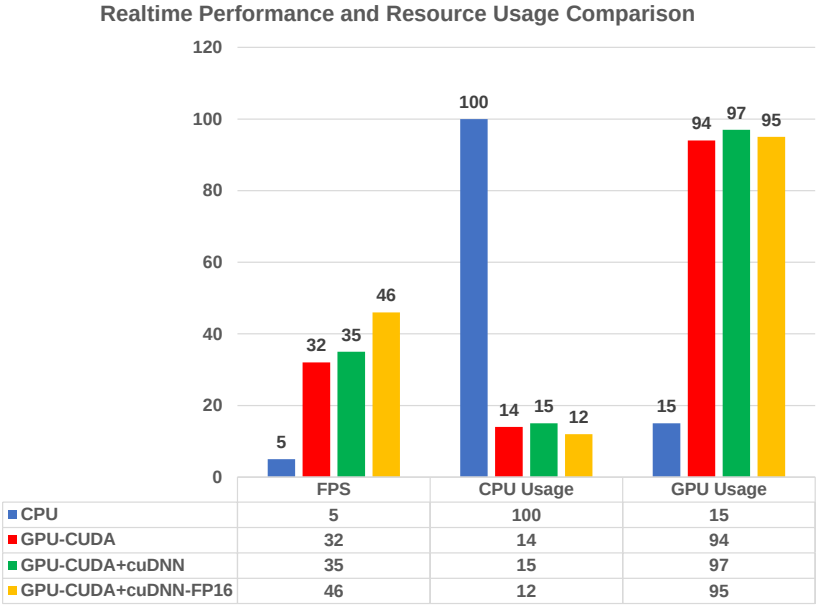


Figure A.4: A comparison of the four variants in terms of FPS achieved and resource usage.

It is important to note that the results here show the **test** runtime performance as opposed to the training runtime performance that was evaluated in the previous section. Nonetheless, the results for variant (1) are in line with findings from the previous section. The CPU implementation of the classifier achieved processing of only **5 FPS** with the resource utilization indicating that the bottleneck was indeed the processing capability of the CPU due to the lack of multiple SMs for parallel compute capability. In addition, the GPU resource consumption confirms that it was indeed unused and hence idle.

On the other hand, variants (2)-(4) utilize the full capacity of the GPU with slight (<2%) variations in the usage attributed to different time of capture of the resource statistics. However, there was a **3 FPS** increase compared to the CUDA based implementation achieved by using the cuDNN library due to the aforementioned parallelization of element-wise matrix multiplications. In contrast, a **significant** increase of **11 FPS** was achieved through the use of tensor core based FP16 computations attributed to the parallelization of element-wise matrix multiplications across multiple kernel strides.

The significant increase in runtime performance by using tensor cores for testing compared to the relatively minimal improvement in runtime during training is particularly due to the effect of using FP16 precision in the training process. Specifically, the reduced precision based computation during training is **detrimental** to training runtime due to possibility of **overshooting** over minima of the loss function leading the model to take longer to converge or meet the specified error tolerance. However, at test time, the runtime is not affected by any error tolerance, and hence the result of kernel stride parallelization is significant.

A.3 Test Environment Setup and Application Stack Integration

A.3.1 OpenCV Build Configuration to Link CUDA and cuDNN

In order for the input images to be processed and imported as tensors that can be transferred to the GPU, the source installation of OpenCV using the following makefile that links the python and numpy executable as well as sets the appropriate flags to build with CUDA and cuDNN for the specific compute architecture (7.5) that supports tensor core computations must be used:

```
cmake   -D CMAKE_BUILD_TYPE=RELEASE \
        -D CMAKE_INSTALL_PREFIX=/usr/local \
        -D OPENCV_EXTRA_MODULES_PATH=/home/npbhatt/opencv_contrib/modules \
        -D BUILD_TIFF=ON \
        -D WITH_GSTREAMER=ON \
        -D WITH_TBB=ON \
        -D BUILD_TBB=ON \
        -D WITH_EIGEN=ON \
        -D WITH_V4L=ON \
        -D WITH_LIBV4L=ON \
        -D WITH_VTK=OFF \
        -D WITH_OPENGL=ON \
        -D WITH_LAPACK=ON \
        -D BUILD_WEBP=OFF \
        -D OPENCV_ENABLE_NONFREE=ON \
        -D INSTALL_C_EXAMPLES=OFF \
        -D PYTHON3_EXECUTABLE=/home/npbhatt/anaconda3/envs/yolov3/bin/python \
        -D PYTHON_INCLUDE_DIR=/home/npbhatt/anaconda3/envs/yolov3/include/python3.6m
     \
        -D PYTHON_LIBRARY=/home/npbhatt/anaconda3/envs/yolov3/lib/libpython3.6m.so \
        -D PYTHON3_NUMPY_INCLUDE_DIRS=/home/npbhatt/anaconda3/envs/yolov3/lib/
    python3.6/site-packages/numpy/core/include \
        -D BUILD_OPENCV_PYTHON3=ON \
        -D INSTALL_PYTHON_EXAMPLES=ON \
        -D BUILD_NEW_PYTHON_SUPPORT=ON \
        -D OPENCV_GENERATE_PKGCONFIG=ON \
        -D BUILD_TESTS=OFF \
        -D BUILD_EXAMPLES=OFF \
        -D WITH_CUDA=ON \
        -D ENABLE_FAST_MATH=ON \
        -D CUDA_FAST_MATH=ON \
        -D WITH_CUDNN=ON \
        -D OPENCV_DNN_CUDA=ON \
        -D CUDA_ARCH_BIN=5.3,6.0,6.1,7.0,7.5 \
        -D CUDA_ARCH_PTX=7.5 \
        -D WITH_CUBLAS=ON ..
```

A.3.2 COCO Dataset - Makefile Flags for CUDA, cuDNN, and FP16 Support

To enable CUDA, cuDNN, and FP16 based tensor core operations, the following options in the makefile file to build the C++ code must be specified:

```
GPU=1 # set GPU=1 to enable CUDA (use version 10.2)
CUDNN=1 # set CUDNN=1 to speedup DNN Ops on GPU (use version 7.6.5)
CUDNN_HALF=1 # set CUDNN_HALF = 1 to enable tensor core Ops
OPENCV=1 # set to 1 to use OpenCV 4.2 for image processing and I/O
AVX=1 # set AVX and OPENMP to 1 to speedup CPU Ops
OPENMP=1
LIBSO=0

```

```
9
10 USE_CPP=0
11 DEBUG=0
12
13 OS := $(shell uname)
14
15 # GeForce RTX 2080 Ti, RTX 2080, RTX 2070, Quadro RTX 8000, Quadro RTX 6000, Quadro
       RTX 5000, Tesla T4, XNOR Tensor Cores
16 ARCH= -gencode arch=compute_75,code=[sm_75,compute_75]
17
18
19 ifeq ($(LIBSO), 1)
20 LIBNAMESO=libdarknet.so
21 APPNAMESO=uselib
22 endif
23
24 ifeq ($(USE_CPP), 1)
25 CC=g++
26 else
27 CC=gcc
28 endif
29
30 CPP=g++ -std=c++11
31 NVCC=nvcc
32 OPTS=-Ofast
33 LDFLAGS= -lm -pthread
34 COMMON= -Iinclude/ -I3rdparty/stb/include
35 CFLAGS=-Wall -Wfatal-errors -Wno-unused-result -Wno-unknown-pragmas -fPIC
36
37 ifeq ($(DEBUG), 1)
38 COMMON+= -DDEBUG
39 CFLAGS+= -DDEBUG
40 else
41 ifeq ($(AVX), 1)
42 CFLAGS+= -ffp-contract=fast -mavx -mavx2 -msse3 -msse4.1 -msse4.2 -msse4a
43 endif
44 endif
45
46 CFLAGS+=$(OPTS)
47
48 ifneq (,$(findstring MSYS_NT,$(OS)))
49 LDFLAGS+=-lws2_32
50 endif
51
52 ifeq ($(OPENCV), 1)
53 COMMON+= -DOPENCV
54 CFLAGS+= -DOPENCV
55 LDFLAGS+= `pkg-config --libs opencv4 2> /dev/null || pkg-config --libs opencv`
56 COMMON+= `pkg-config --cflags opencv4 2> /dev/null || pkg-config --cflags opencv`
57 endif
58
59 ifeq ($(OPENMP), 1)
60 CFLAGS+= -fopenmp
61 LDFLAGS+= -lgomp
62 endif
63
64 ifeq ($(GPU), 1)
```

```
COMMON+= -DGPU -I/usr/local/cuda/include/
CFLAGS+= -DGPU
ifeq ($(OS),Darwin) #MAC
LDFLAGS+= -L/usr/local/cuda/lib -lcuda -lcudart -lcublas -lcurand
else
LDFLAGS+= -L/usr/local/cuda/lib64 -lcuda -lcudart -lcublas -lcurand
endif
endif

ifeq ($(CUDNN), 1)
COMMON+= -DCUDNN
ifeq ($(OS),Darwin) #MAC
CFLAGS+= -DCUDNN -I/usr/local/cuda/include
LDFLAGS+= -L/usr/local/cuda/lib -lcudnn
else
CFLAGS+= -DCUDNN -I/usr/local/cudnn/include
LDFLAGS+= -L/usr/local/cudnn/lib64 -lcudnn
endif
endif

ifeq ($(CUDNN_HALF), 1)
COMMON+= -DCUDNN_HALF
CFLAGS+= -DCUDNN_HALF
ARCH+= -gencode arch=compute_70,code=[sm_70,compute_70]
endif
```

Appendix B

MPC-PF: Additional Qualitative Illustrations

B.1 Potential Field Illustrations

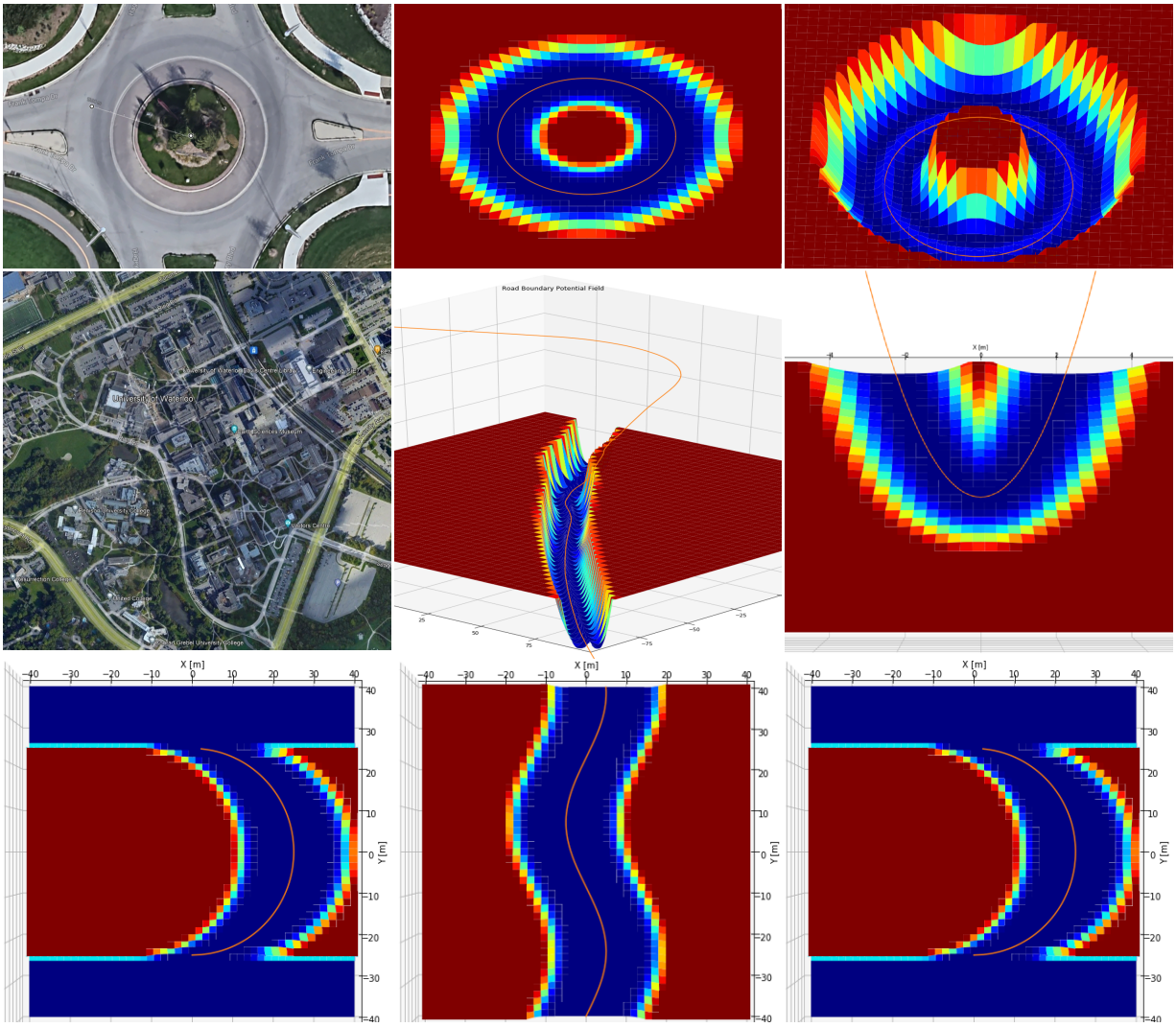


Figure B.1: Illustration of several potential fields generated for various road geometries.

B.2 Extracting Traversable Trajectories for Multimodal Prediction

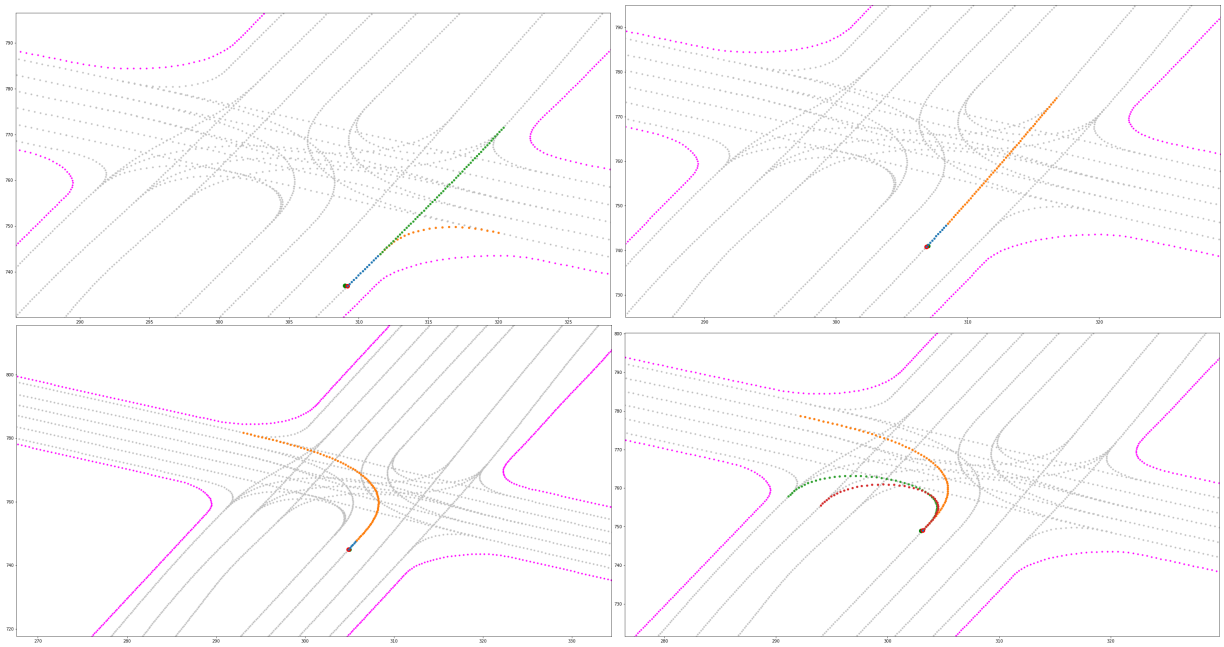


Figure B.2: Illustration of all traversable trajectories given different initial positions of the SDV on lower side of intersection.

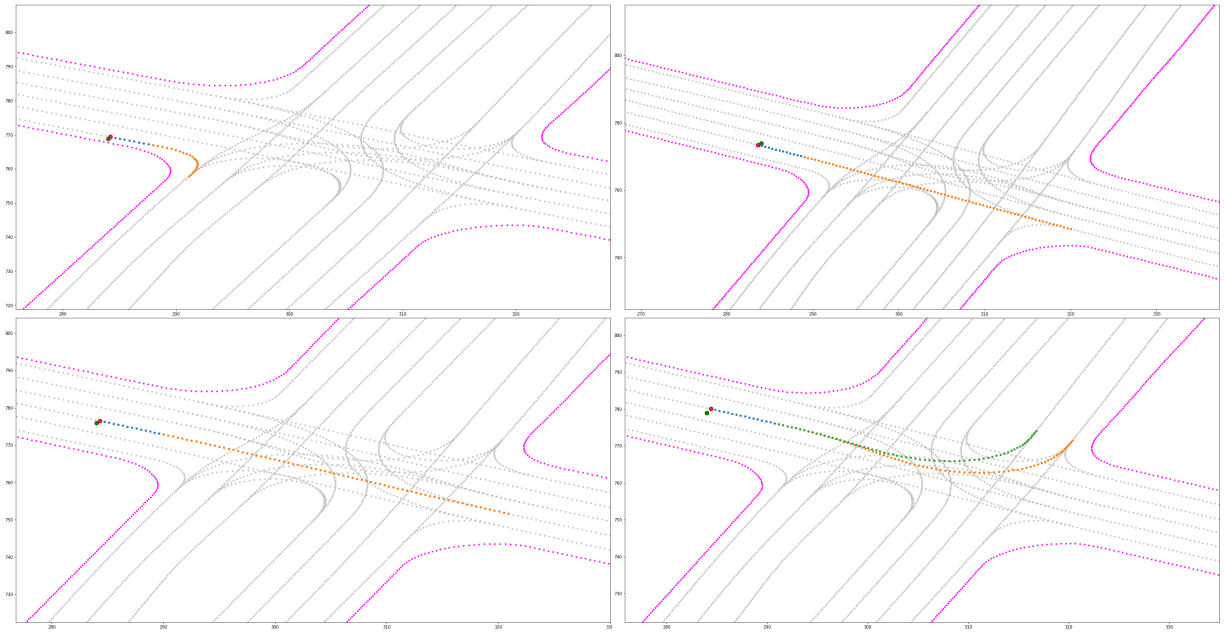


Figure B.3: Illustration of all traversable trajectories given different initial positions of the SDV on left side of intersection.

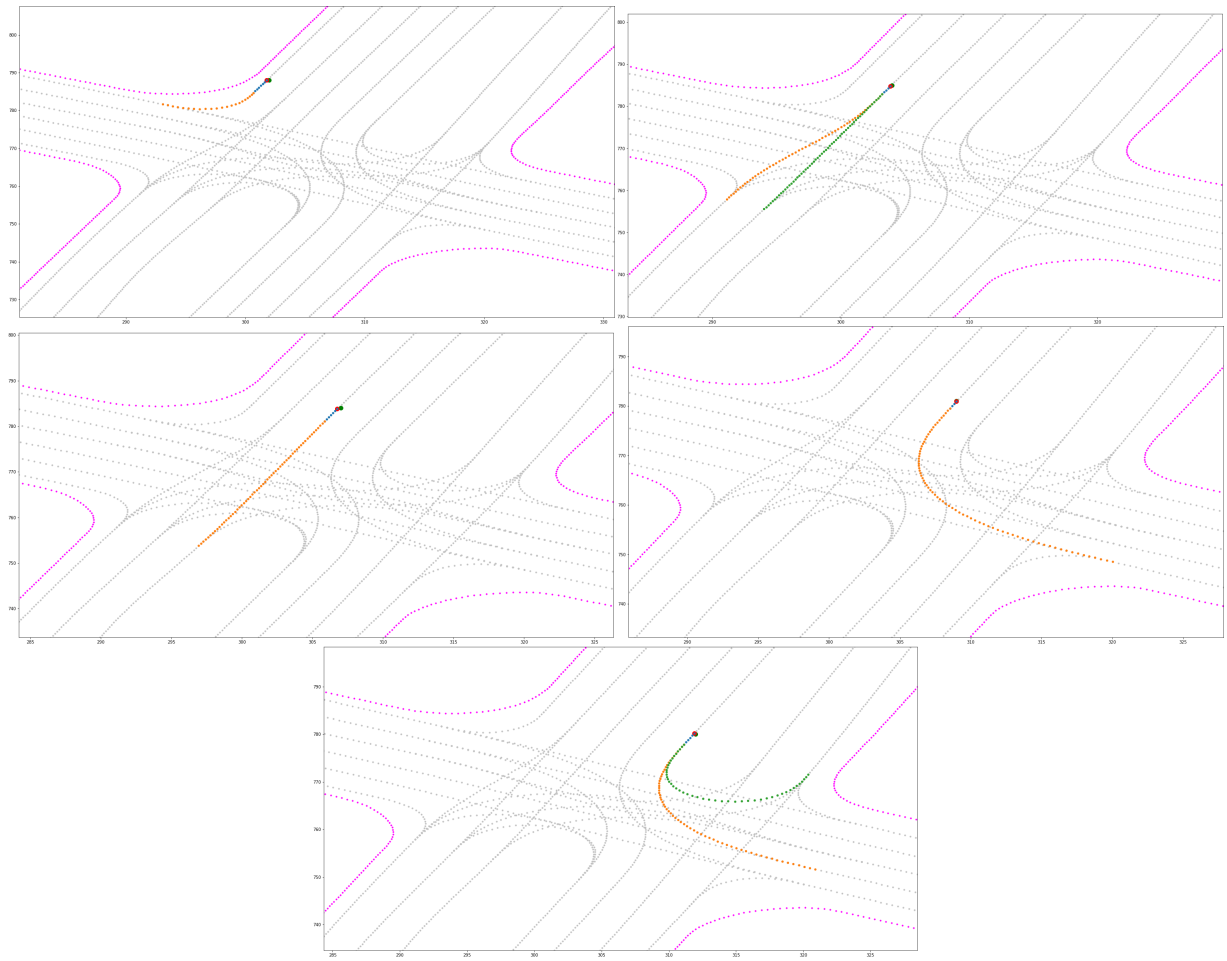


Figure B.4: Illustration of all traversable trajectories given different initial positions of the SDV on upper side of intersection.

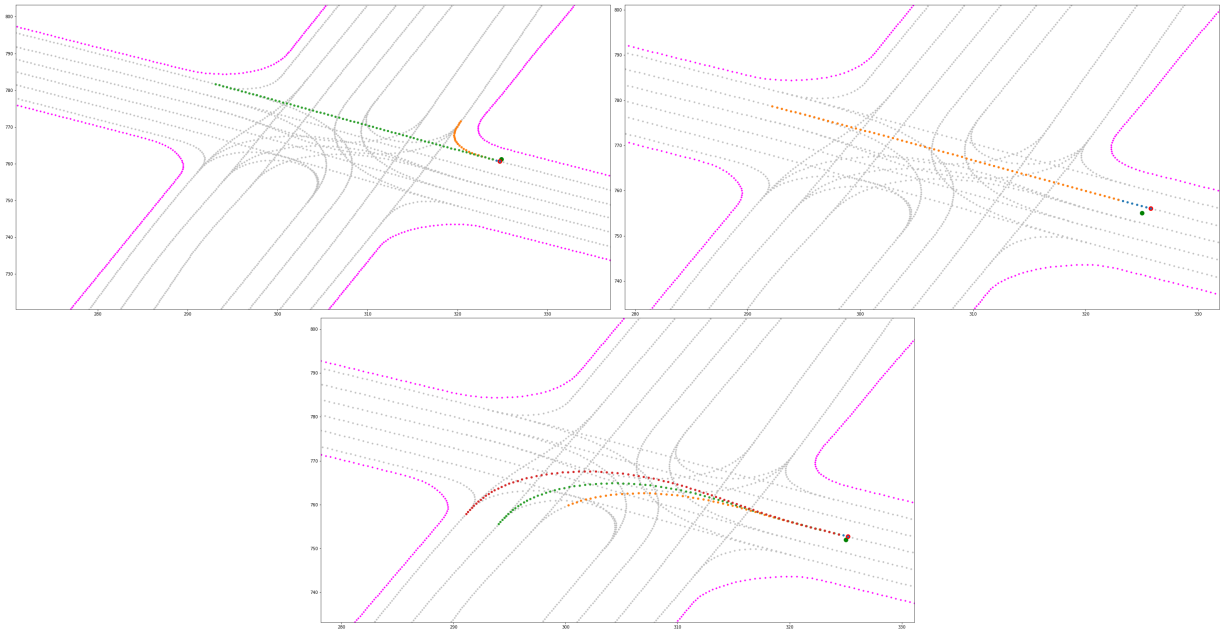


Figure B.5: Illustration of all traversable trajectories given different initial positions of the SDV on right side of intersection.

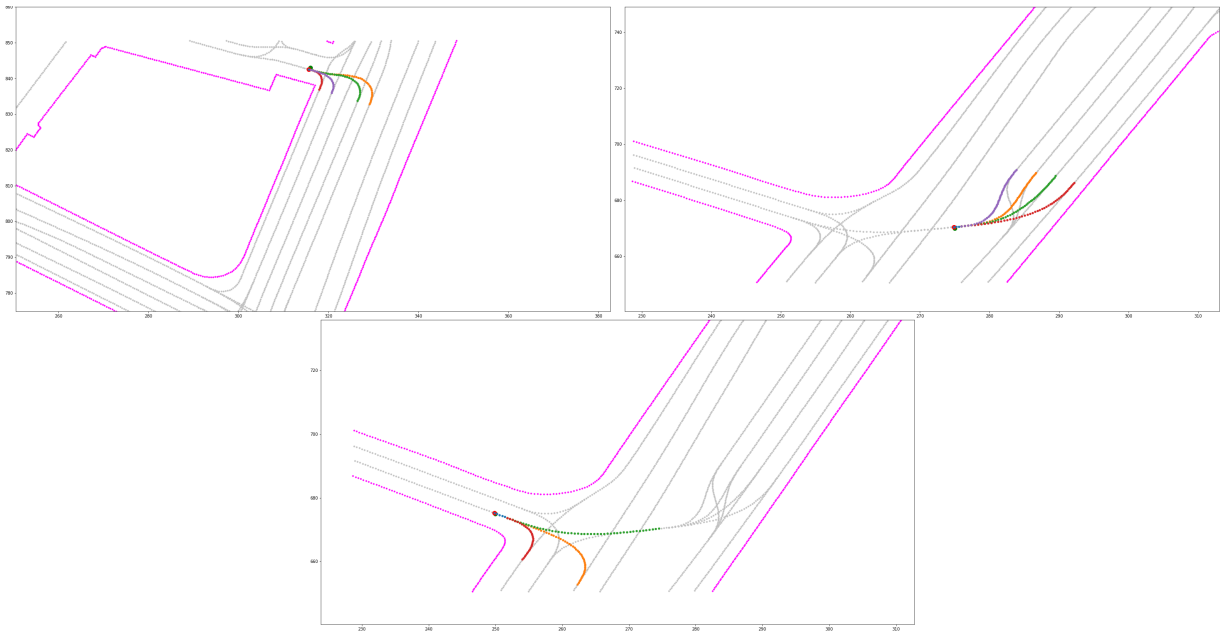


Figure B.6: Illustration of all traversable trajectories given different initial positions of the SDV for merging into another intersection.

Appendix C

Experimental Setup - Additional Details

C.1 WATonoBus Vehicle Parameters

Table C.1: Vehicle and Tire Parameters

Parameter	Unit	Value	Description
m, m_s	kg	2260, 1989	Total/sprung mass
I_z, I_w	kg.m^2	4650, 1.68	Vehicle/wheel moment of inertia
L_f, L_r	m	1.62, 1.56	Front/rear track width

C.2 WATonoBus Sensors, Setup, and Operation

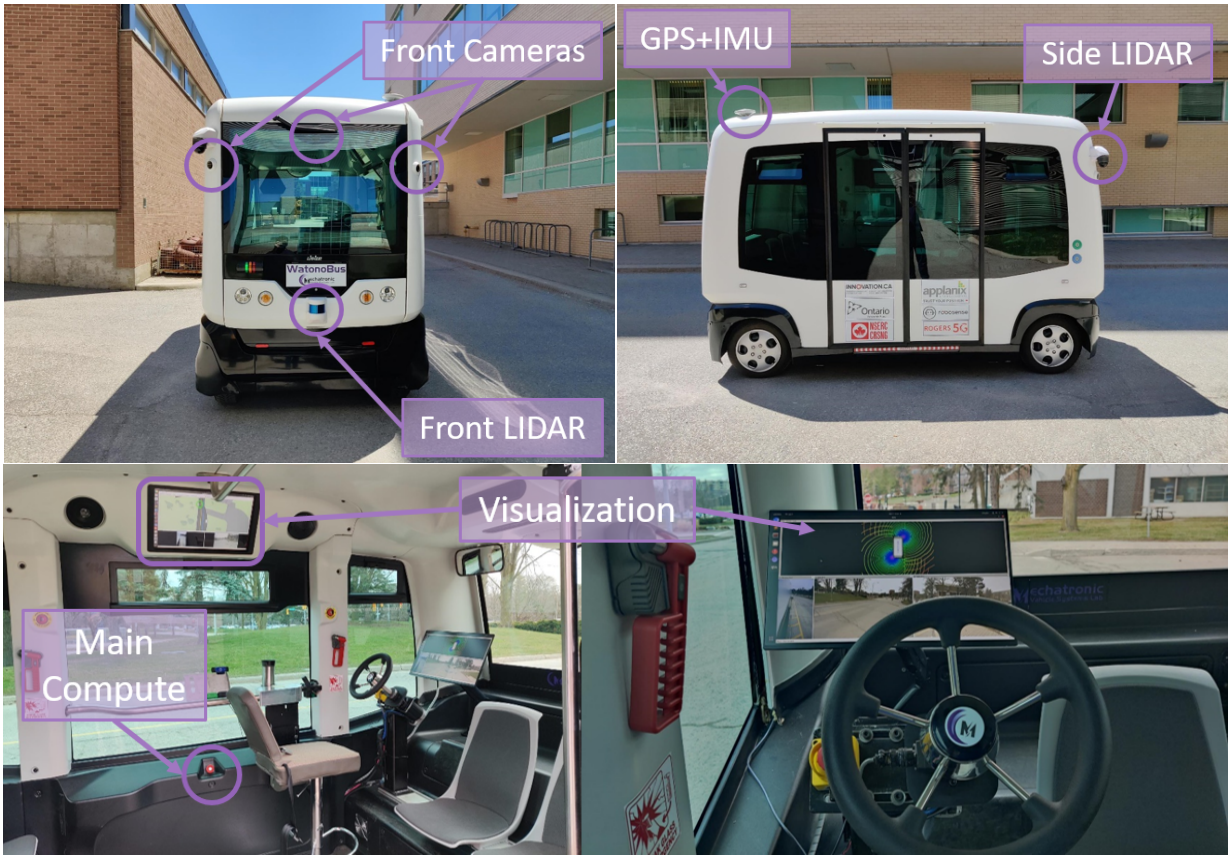


Figure C.1: Illustration of the WATonoBus sensor suite and visualization utilities that the author of the thesis led developments on.

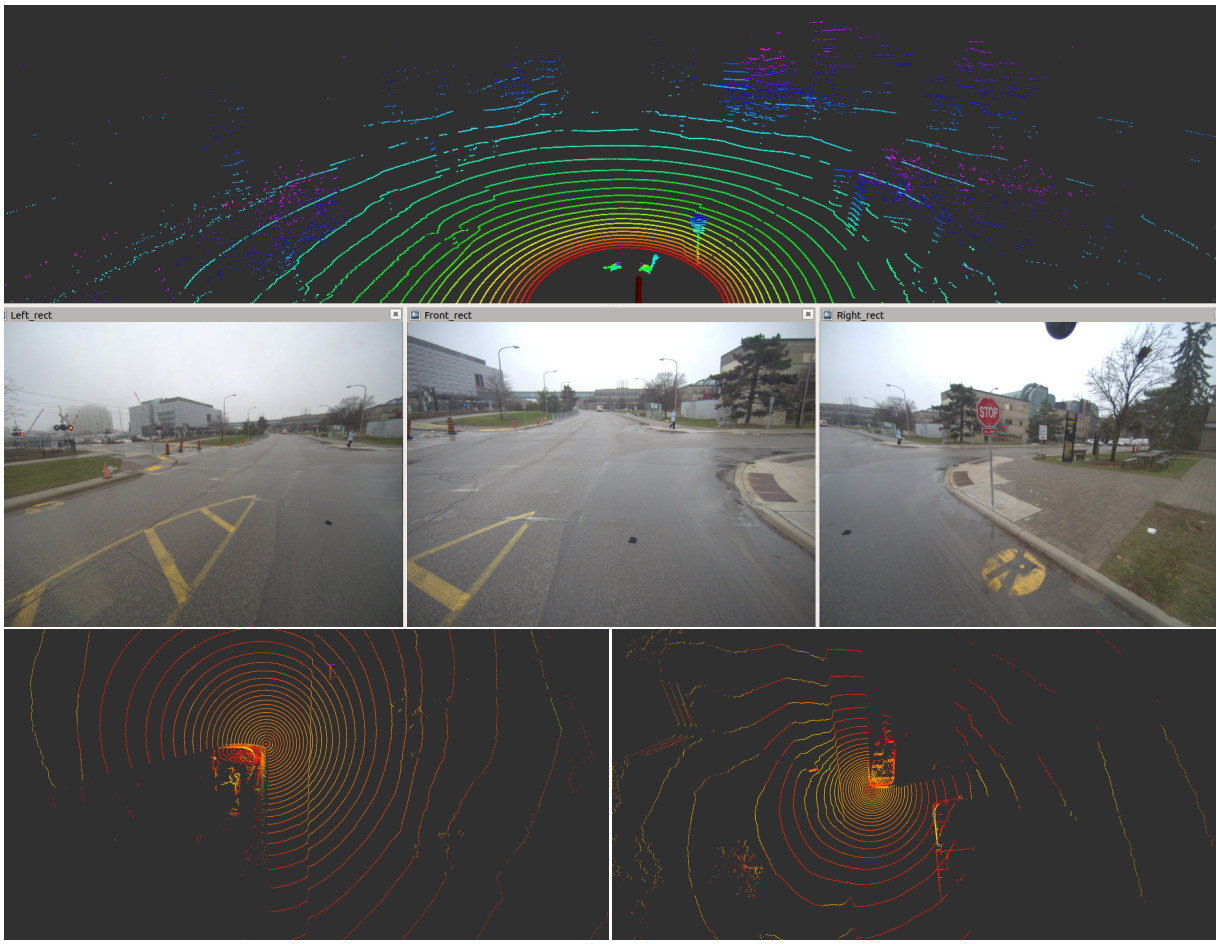


Figure C.2: Illustration of the WATonoBus sensor data from front cameras and LIDAR along with data from side LIDARs.

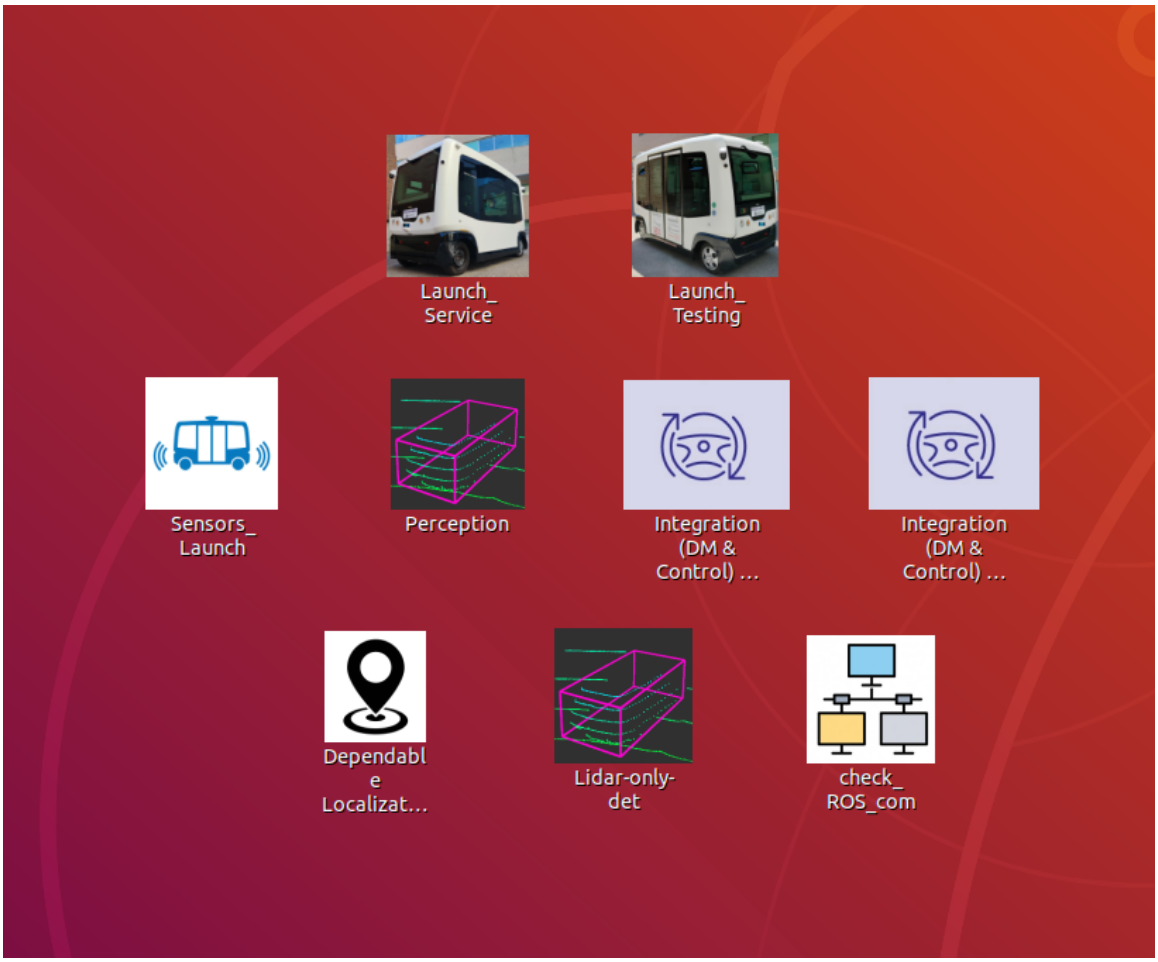


Figure C.3: Illustration of the WATonoBus software suite setup and auto launch utilities that the author of the thesis led developments on.

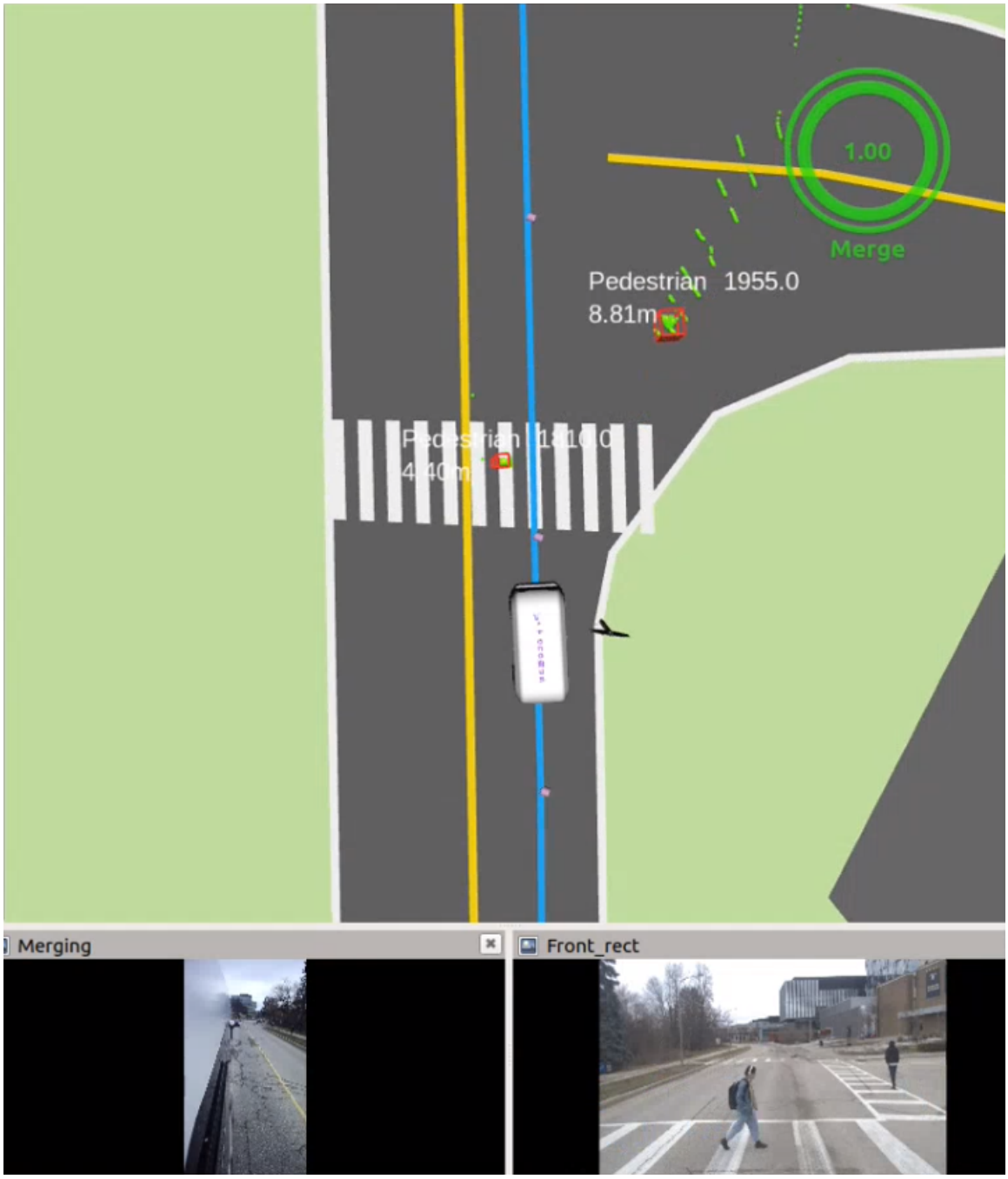


Figure C.4: Illustration of the WATonoBus at an intersection with pedestrians crossing. The nominal path is shown in blue and detections in red.

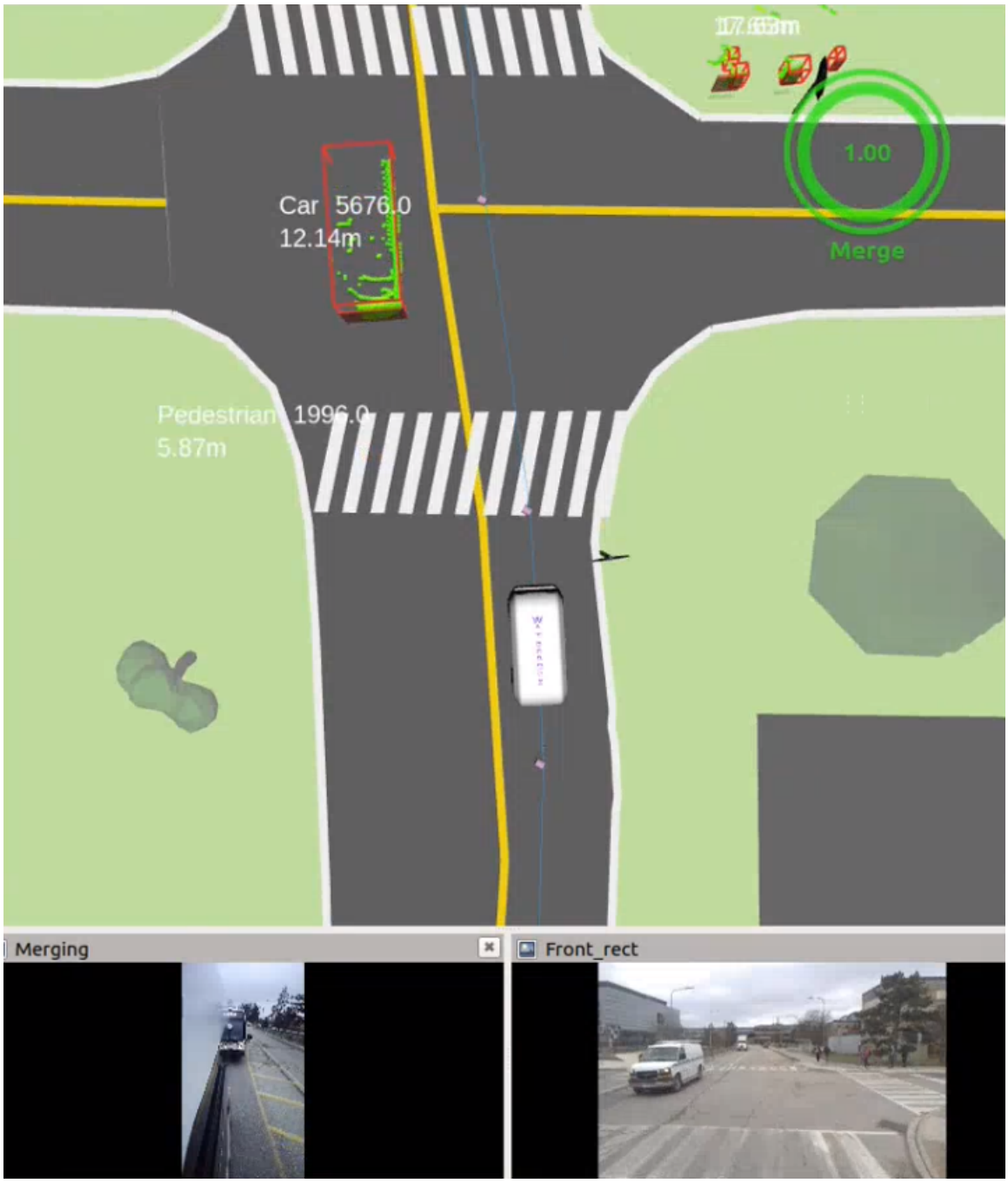


Figure C.5: Illustration of the WATonoBus at an another intersection with a car on the oncoming lane and pedestrians at the crossing.

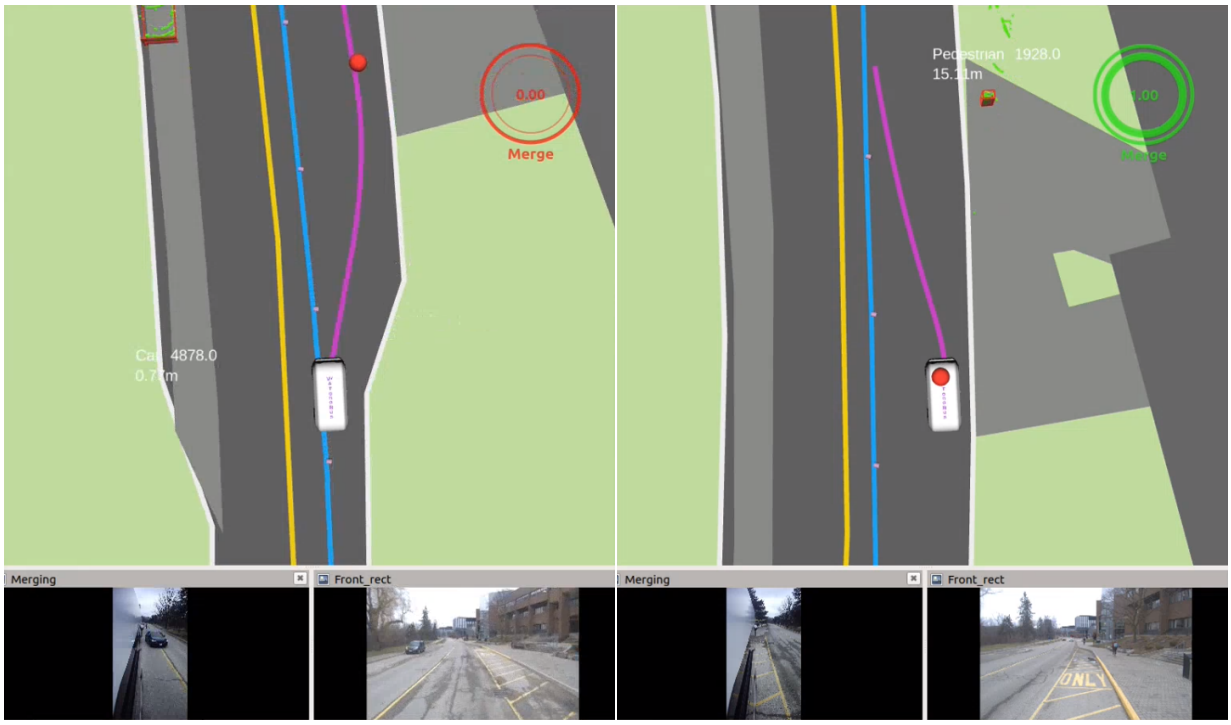


Figure C.6: Illustration of the WATonoBus pulling over to a bus stop and subsequently merging back onto the main lane. The nominal path is shown in yellow and the planned path in magenta.

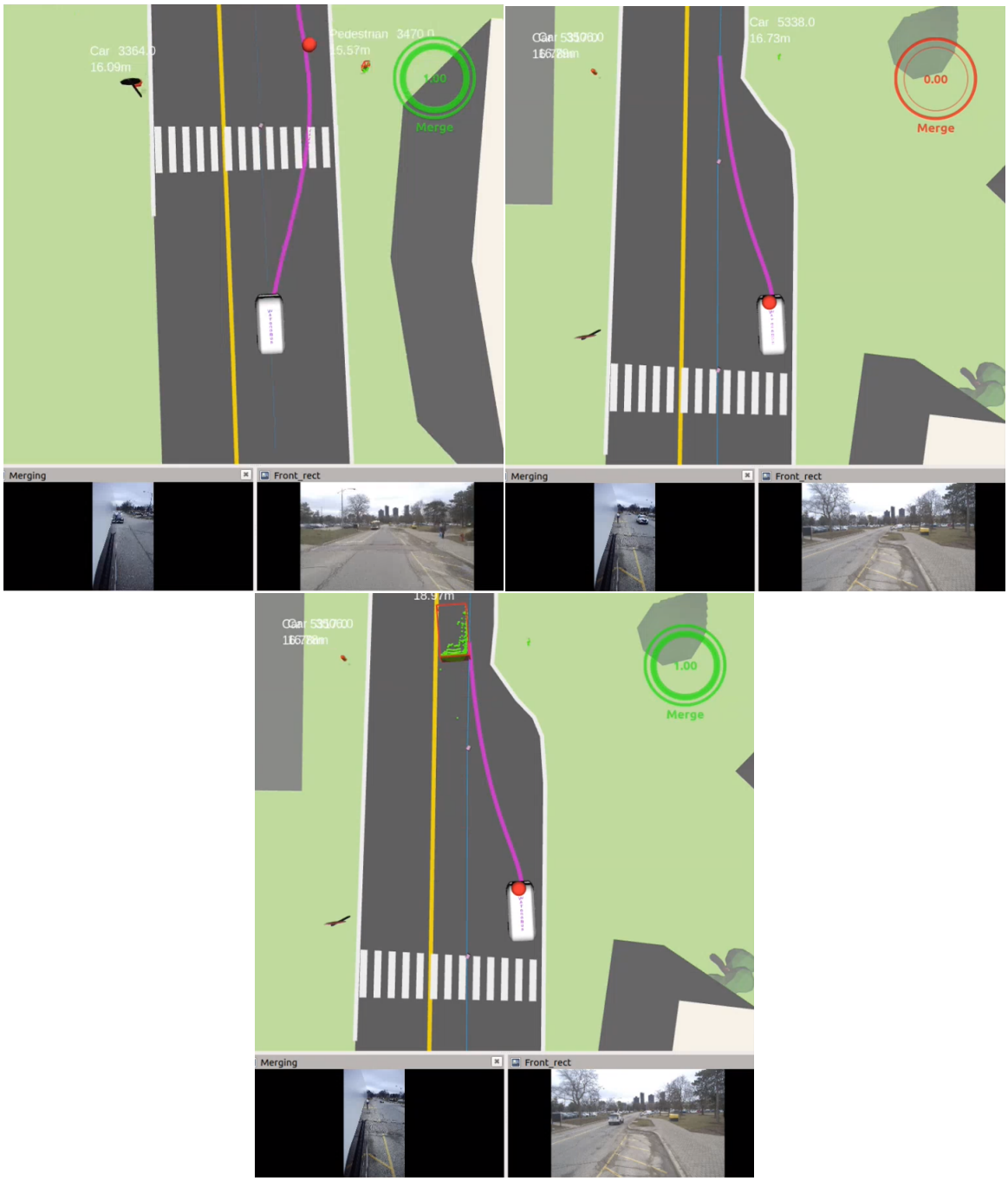


Figure C.7: Illustration of the WATonoBus pulling over to another bus stop and subsequently waiting for a vehicle passing by on the main lane before merging back.