Delft University of Technology

Interaction-Aware Motion Planning in Crowded Dynamic Environments

Ferreira de Brito, B.F.

DOI
10.4233/uuid:0bc3677d-dc15-4f93-a079-8d600e967c5a

Publication date
2022

Document Version
Final published version

Citation (APA)
Ferreira de Brito, B. F. (2022). *Interaction-Aware Motion Planning in Crowded Dynamic Environments*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:0bc3677d-dc15-4f93-a079-8d600e967c5a

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Interaction-Aware Motion Planning in Crowded Dynamic Environments

Bruno Brito

Interaction-Aware Motion Planning in Crowded Dynamic Environments

Bruno Brito

Interaction-Aware Motion Planning in Crowded Dynamic Environments

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen
chair of the Board for Doctorates
to be defended publicly on
Thursday 6 October 2022 at 15:00 o'clock

by

Bruno Filipe Ferreira de BRITO

Master of Electrical and Computers Engineering,
Faculdade de Engenharia da Universidade do Porto, Portugal,
born in Porto, Portugal.

This dissertation has been approved by the promoters.
Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof.dr. R. Babuska	Delft University of Technology, promotor
Dr. J. Alonso-Mora	Delft University of Technology, promotor
Independent members:	
Prof.dr. D. M. Gavrila	Delft University of Technology
Prof.dr.ir. C. G. Chorus	Delft University of Technology
Prof.dr. J. P. How	Massachusetts Institute of Technology
Prof.dr. K. V. Hindriks	Vrije Universiteit Amsterdam
Prof.dr. M. Wang	Technische Universität Dresden

The research described in this thesis was partly supported by the Netherlands Organization for Scientific Research (NWO) domain Applied Sciences.

Published and distributed by: Bruno Brito
E-mail: brunoffbrito0@gmail.com

Keywords: Motion planning, machine learning, interaction, collision avoidance, autonomous vehicles, mobile robots, trajectory prediction, decision-making, dynamic environments, navigation among humans
Front & back: ProefschriftMaken

Printed by: ProefschriftMaken

Copyright © 2022 by Bruno Brito

ISBN 978-94-6366-605-3

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission of the author.

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

To my family.

Contents

Summary **xi**

Samenvatting **xv**

Acknowledgments **xix**

1 Introduction **1**

1.1 Motivation 2
1.2 Approach 3
    1.2.1 Local Motion Planning 3
    1.2.2 Trajectory Prediction. 4
    1.2.3 Learning for Global Guidance 5
1.3 Contributions and Outline 6
1.4 Notation 8

2 Background **9**

2.1 Model Predictive Control. 10
2.2 Supervised Learning 11
    2.2.1 Variational Bayes. 13
2.3 Reinforcement Learning 13
    2.3.1 Proximal Policy Optimization 14
    2.3.2 Soft Actor-Critic 16

3 Literature Review **19**

3.1 Motion Planning in Dynamic Environments 20
    3.1.1 Mobile Robots 20
    3.1.2 Autonomous Driving. 21
3.2 Trajectory Prediction. 23
    3.2.1 Traditional Approaches 23
    3.2.2 Deep Learning 23
3.3 Learning Global Guidance Policies 24
    3.3.1 Navigation Among Crowds 24
    3.3.2 Learning-Enhanced MPC. 25
    3.3.3 Combining MPC with RL. 25
3.4 Interaction-aware Motion Planning 26
    3.4.1 Traditional Methods 26
    3.4.2 Search-based Methods 26
    3.4.3 Optimization-based Methods 27
    3.4.4 Game Theoretic Methods. 27
    3.4.5 Learning-based Methods 27

3.5 Conclusions 28

4 **Model Predictive Contouring Control for Collision Avoidance in Dynamic Environments** **29**
4.1 Introduction 30
4.2 Preliminaries 31
  4.2.1 Robot Description 31
  4.2.2 Static Obstacles 31
  4.2.3 Dynamic Obstacles 32
  4.2.4 Global Reference Path 32
  4.2.5 Problem Formulation 32
4.3 Method 33
  4.3.1 Static Collision Avoidance 33
  4.3.2 Dynamic Collision Avoidance 34
  4.3.3 Model Predictive Contouring Control 36
4.4 Results - Mobile Robot 39
  4.4.1 Experimental Setup 39
  4.4.2 Parameter Evaluation 40
  4.4.3 Static Collision Avoidance 41
  4.4.4 Dynamic Collision Avoidance 42
4.5 Results - Autonomous Car 46
4.6 Conclusions and Future Work 51

5 **Social-VRNN: One-Shot Multi-modal Trajectory Prediction for Interacting Pedestrians** **53**
5.1 Introduction 54
5.2 Variational Recurrent Neural Network 56
  5.2.1 Multi-modal Trajectory Prediction Problem Formulation 56
  5.2.2 Input Feature Extraction Module 58
  5.2.3 Probabilistic Inference Module 58
  5.2.4 Multi-modal Trajectory Prediction Distribution 59
  5.2.5 Improving Diversity 59
  5.2.6 Training Procedure 60
5.3 Experiments 61
  5.3.1 Experimental Settings 61
  5.3.2 Performance Evaluation 61
  5.3.3 Qualitative Analysis 63
5.4 Conclusions 66

6 **Where to go next: Learning a Subgoal Recommendation Policy for Navigation Among Pedestrians** **67**
6.1 Introduction 68
6.2 Preliminaries 69
  6.2.1 Problem Formulation 69
  6.2.2 Agent Dynamics 70
  6.2.3 Modeling Other Agents' Behaviors 70

6.3 Method. 71
6.3.1 Learning a Subgoal Recommender Policy. 71
6.3.2 Local Collision Avoidance 73
6.3.3 PPO-MPC 74
6.4 Results 76
6.4.1 Experimental Setup 76
6.4.2 Training Procedure. 77
6.4.3 Ablation Study 77
6.4.4 Qualitative Analysis 80
6.4.5 Performance Results 83
6.5 Conclusions and Future Work 85

7 Learning Interaction-Aware Guidance for Trajectory Optimization in Dense Traffic Scenarios 87
7.1 Introduction 88
7.2 Problem Formulation. 91
7.3 Interactive Model Predictive Control 92
7.3.1 Interactive Planner 92
7.3.2 Local Motion Planner 93
7.3.3 Training Procedure. 95
7.3.4 Online Planning 96
7.4 Modeling Other Traffic Drivers' Behaviors 97
7.5 Experiments 99
7.5.1 Experimental Setup 99
7.5.2 Driving Scenarios 100
7.5.3 Evaluation Scenarios 102
7.5.4 Evaluation Metrics 102
7.5.5 Training Procedure. 103
7.5.6 Qualitative Results 104
7.5.7 Quantitative Results 107
7.5.8 Performance Analysis 110
7.5.9 Discussion 115
7.6 Conclusions 115

8 Conclusions and Future Work 117
8.1 Conclusions 118
8.2 Future Work 121
8.2.1 Interaction-Aware Motion Planning 121
8.2.2 Learning Guidance Policies. 121
8.2.3 Constraints-Aware Learning 122
8.2.4 Continual Learning 122

A The Jackal 123

B The Toyota Prius 125

Bibliography 127

Curriculum vitæ **147**

List of publications **149**

Summary

Autonomous robots will profoundly impact our society, making our roads safer, reducing labor costs and carbon dioxide (CO2) emissions, and improving our life quality. However, to make that happen, robots need to navigate among humans, which is extremely difficult. Firstly, humans do not explicitly communicate their intentions and use intuition to reason about others' plans to avoid collisions. Secondly, humans exploit interactions to navigate efficiently in cluttered environments. Traditional motion planning methods for autonomous navigation in human environments use geometry, physics, topologies, and handcrafted functions to account for interaction but only plan one step. In contrast, trajectory optimization methods allow planning over a prediction horizon accounting for the environment evolution. Yet, these methods scale poorly with the number of agents and assume structured scenarios with a limited number of interacting agents. Learning-based approaches overcome the latter by learning a policy's parameters offline, e.g., from data or simulation. However, to date, learned policies show poor performance and unpredictable behavior when employed in reality as the conditions differ from the learning environment. Moreover, learning-based approaches do not guarantee collision avoidance or feasibility with respect to the robot dynamics. Therefore, this thesis aims to develop motion planning algorithms generating online predictive and interaction-aware motion plans to enable robots' safe and efficient navigation among humans.

The first main contribution of this thesis is a predictive motion planning algorithm for autonomous robot navigation in unstructured environments populated with pedestrians. The proposed method builds on nonlinear model-predictive contouring control proposing a local formulation (LMPCC) to generate predictive motion plans in real-time. Static collision avoidance is achieved by constraining the robot's positions to stay within a set of convex regions approximating the surrounding free space computed from a static map. Moreover, an upper bound for the Minkowski sum of a circle and an ellipse is proposed and used as an inequality constraint to ensure dynamic collision avoidance, assuming the robot's space as a circle and the dynamic obstacles', for instance pedestrians, space ellipsoid. The LMPCC approach is analyzed and compared against a reactive and a learning-based approach in simulation. Experimentally, the method is tested fully onboard on a mobile robot platform (Clearpath Jackal) and in an autonomous car (Toyota Prius).

In real scenarios, pedestrians do not explicitly communicate their intentions, and therefore, LMPCC uses a constant velocity (CV) model to estimate their future trajectories. However, CV predictions ignore the environment constraints, e.g., static obstacles, the interaction between agents, and the inherent uncertainty and multimodality of the pedestrians' motion. Hence, this thesis presents a variational recurrent neural network architecture (Social-VRNN) for interaction-aware and multi-modal trajectory predictions.

The Social-VRNN fuses information of the pedestrian's dynamics, static obstacles, and surrounding pedestrians and outputs the parameters of a Gaussian Mixture Model (GMM). A variational Bayesian learning approach is employed to learn the model's parameters minimizing the evidence lower bound (ELBO). Experimental results on real and simulation data are presented, showing that our model can effectively learn to predict multiple trajectories capturing the different courses that a pedestrian may follow.

Enhancing the LMPCC method with interaction-aware predictions is insufficient to enable safe and efficient autonomous navigation in cluttered environments. The LMPCC is a local trajectory optimization method and considers a limited planning horizon to enable online motion planning. Consequently, LMPCC plans can be locally optimal, which may result in catastrophic failures, such as deadlocks and collisions, in the long term. To overcome the latter, global guidance, e.g., cost-to-go heuristics, to the optimization problem is an option. In contrast to optimization-based methods, learning-based methods, i.e., deep reinforcement learning (DRL), allow learning policies to optimize long-term rewards in an offline training phase. Therefore, this thesis introduces two novel frameworks enhancing state-of-art online optimization-based planners with learned global guidance policies applied to mobile robot navigation in cluttered environments and autonomous vehicles driving in dense traffic.

Firstly, the Goal-Oriented Model Predictive Controller (GO-MPC) is introduced, tackling the problem that the robot's global goal is often located far beyond the planning horizon resulting in locally optimal motion plans. The framework proposes to use DRL to learn an interaction-aware policy providing the next optimal subgoal position to an MPC planner. The recommended subgoal helps the robot progress towards its end goal and accounts for the expected interaction with other agents. Based on the recommended subgoal, the MPC planner then optimizes the inputs for the robot, satisfying its kinodynamic and collision avoidance constraints. Simulation results are presented demonstrating that GO-MPC enhances the navigation performance in terms of safety and efficiency, i.e., travel time, compared to solely based MPC and deep RL frameworks in mixed settings, i.e., with cooperative and non-cooperative agents, and multi-robot scenarios.

Secondly, the interactive Model Predictive Controller (IntMPC) for safe navigation in dense traffic scenarios is presented. While GO-MPC learns a subgoal policy, the IntMPC learns a velocity reference policy exploring the connection between human driving behavior and their velocity changes when interacting. Hence, the IntMPC approach learns, via deep Reinforcement Learning (RL), an interaction-aware policy providing global guidance as a velocity reference allowing to control and exploit the interaction with the other vehicles. Simulation results are presented demonstrating that the learned policy can reason about the cooperativeness of other vehicles and enable the local planner with interactive behavior to pro-actively merge in dense traffic while remaining safe in case the other vehicles do not yield.

Overall, this thesis contributes to enhancing autonomous robots with predictive behavior, with the ability to infer the others' trajectories and operate in cluttered environments.

However, two important limitations remain to be solved: the proposed motion planner

computes open-loop interaction-aware motion plans and does not account for interaction in closed-loop. Moreover, the prediction model and guidance policies rely solely on offline learning. Future works may investigate how to account for interaction in the planning stage and how online data streams can be used to improve the navigation algorithm's performance over time.

Samenvatting

Autonome robots zullen een grote impact hebben op onze samenleving, onze wegen veiliger maken, de arbeidskosten en de uitstoot van koolstofdioxide (CO2) verlagen en onze levenskwaliteit verbeteren. Om dat mogelijk te maken, moeten robots echter veilig kunnen navigeren in de nabijheid van mensen, wat op het moment een grote uitdaging is. Ten eerste communiceren mensen niet expliciet hun bedoelingen en gebruiken ze hun intuïtie (in plaats van regels) om andere mensen te ontwijken. Ten tweede maken mensen gebruik van interacties om efficiënt te navigeren in drukke omgevingen. Traditionele bewegingsplanningsmethoden voor autonome navigatie in menselijke omgevingen maken gebruik van geometrie, fysica, topologieën en handgemaakte functies om rekening te houden met interactie. Daarentegen maken trajectoptimalisatiemethoden het mogelijk om beweging over een voorspellingshorizon te plannen, rekening houdend met de veranderingen in de omgeving. Deze methoden schalen echter slecht met het aantal andere robots en/of mensen (ook wel “agenten”) en gaan ze uit van gestructureerde scenario’s met een beperkt aantal interacterende agenten. Op deze vlakken doen methodes die op zelf-lerende algoritmes gebaseerd zijn het beter, omdat de parameters die het beleid bepalen offline geleerd kunnen worden, bijvoorbeeld vanuit sensor of simulatie data. Tot op heden presteren deze methodes echter slecther en vertonen ze onvoorspelbaar gedrag wanneer het in de praktijk wordt toegepast, aangezien de omstandigheden verschillen van de leeromgeving. Bovendien bieden zelf-lerende beweginsplanners geen garantie voor het vermijden van botsingen of haalbaarheid met betrekking tot de robotdynamiek. Daarom heeft dit proefschrift tot doel bewegingsplanningsalgoritmen te ontwikkelen die online voorspellende en interactiebewuste bewegingsplannen genereren om veilige en efficiënte navigatie van robots rond mensen mogelijk te maken.

De eerste belangrijke bijdrage van dit proefschrift is een voorspellend bewegingsplanningsalgoritme voor autonome robotnavigatie in ongestructureerde omgevingen met voetgangers. De voorgestelde methode bouwt voort op nonlinear Model Predictive Contouring Control en stelt een lokale formulering (LMPCC) voor om in realtime voorspellende bewegingsplannen te genereren. Het vermijden van botsingen met statische objecten wordt bereikt door de geplande posities van de robot te beperken binnen een reeks convexe gebieden die de vrije ruimte benaderen. Deze gebieden worden berekend op basis van een statische kaart van de omgeving. Bovendien wordt een bovengrens voor de Minkowski-som van een cirkel en een ellips voorgesteld en gebruikt als een ongelijkheidsbeperking om te garanderen dat botsingen met dynamicshe objecten worden vermeden, ervanuitgaande dat de robot een circkelvormige ruimte inneemt en de objecten (bijvoorbeeld voetgangers) ellipsoïde vormig zijn. De LMPCC-methode wordt geanalyseerd en vergeleken met een reactieve, zelf-lerende methode in simulatie. Experimenteel wordt de methode volledig aan boord getest op een mobiel robotplatform (Clearpath Jackal) en in een autonome auto

(Toyota Prius).

In de werkelijkheid communiceren voetgangers niet expliciet hun intenties, en daarom gebruikt LMPCC een model met constante snelheid (CV) om hun toekomstige trajecten te schatten. CV-voorspellingen negeren echter de omgevingsbeperkingen (bijvoorbeeld statische obstakels), de interactie tussen agenten en de inherente onzekerheid en multimodaliteit van de beweging van voetgangers. Daarom presenteert dit proefschrift een Variational Recurrent Neural Network architectuur (Social-VRNN) voor interactiebewuste en multimodale trajectvoorspellingen. De Social-VRNN combineert informatie over de dynamiek van de voetganger, statische obstakels en omringende voetgangers en geeft de parameters weer als een Gaussian Mixture Model (GMM). Er wordt een variatieve Bayesiaanse leerbenadering gebruikt om de parameters van het model te leren, waarbij de ondergrens van het bewijs (ELBO) wordt geminimaliseerd. Experimentele resultaten op echte data en simulatie data worden gepresenteerd, wat aantoont dat ons model effectief kan leren meerdere trajecten te voorspellen die de verschillende paden van een voetganger omvatten.

Het verbeteren van de LMPCC-methode met interactiebewuste voorspellingen is onvoldoende om veilige en efficiënte autonome navigatie in onoverzichtelijke omgevingen mogelijk te maken. De LMPCC is een lokale trajectoptimalisatiemethode en houdt rekening met een beperkte planningshorizon om online bewegingsplanning mogelijk te maken. Als resultaat daarvan kunnen LMPCC-plannen lokaal optimaal zijn, wat op lange termijn kan leiden tot catastrofale storingen, zoals volledige stilstand en botsingen. Globale begeleiding voor het optimalisatieprobleem, bijv. via een cost-to-go-heuristiek, is een optie om dit probleem op te lossen. In tegenstelling tot standaard optimalisatie methoden, maken zelf-lerende methoden zoals deep enhancement learning (DRL) het leerbeleid mogelijk om langetermijnbeloningen te optimaliseren in een offline trainingsfase. Daarom introduceert dit proefschrift twee nieuwe methoden die optimalisatie planners verbeteren met een geleerde globale begeleiding die wordt toegepast op mobiele robotnavigatie in drukke omgevingen en autonome voertuigen die in druk verkeer rijden.

Ten eerste wordt de Goal-Oriented Model Predictive Controller (GO-MPC) geïntroduceerd, waarmee het probleem wordt aangepakt dat het globale doel van de robot zich vaak ver buiten de planningshorizon bevindt, wat resulteert in lokaal optimale bewegingsplannen. Deze methode stelt voor om DRL te gebruiken om interactiebewust te leren en de volgende optimale subdoelpositie door te sturen naar de MPC-planner. Het aanbevolen subdoel helpt de robot om zijn einddoel te bereiken en houdt rekening met de verwachte interactie met andere agenten. Op basis van het aanbevolen subdoel optimaliseert de MPC-planner vervolgens de besturingssignalen van de robot, waarbij de kinodynamiek wordt gerespecteerd en botsingen worden vermeden. Er worden simulatieresultaten gepresenteerd die aantonen dat GO-MPC de navigatieprestaties verbetert op het gebied van veiligheid en efficiëntie, d.w.z. reistijd, in vergelijking met standaard MPC en diepe RL-methodes in gemengde omgevingen, d.w.z. met coöperatieve en niet-coöperatieve agenten, en in multi-robot scenario's.

Ten tweede wordt de interactieve Model Predictive Controller (IntMPC) voor veilige navigatie in drukke verkeers scenario's gepresenteerd. Terwijl GO-MPC een subdoelbeleid leert, leert de IntMPC een snelheidsreferentiebeleid dat het verband onderzoekt tussen menselijk rijgedrag en hun snelheidsveranderingen tijdens interactie. Daarom leert de IntMPC-benadering, via diepgaande Reinforcement Learning (RL), een interactiebewust beleid dat globale begeleiding aanbiedt als snelheidsreferentie waarmee de interactie met de andere voertuigen wordt gecontrolleerd. Er worden simulatieresultaten gepresenteerd die aantonen dat het geleerde beleid kan redeneren over de coöperatie van andere voertuigen en de lokale planner in staat stelt met interactief gedrag proactief in te voegen in druk verkeer terwijl het voertuig ook veilig beweegt als de andere voertuigen geen ruimte geven.

Dit proefschrift draagt in het algemeen bij aan het verbeteren van autonome robots door hun gedrag voorspelbaar te maken, hun vermogen om de trajecten van anderen af te leiden te verbeteren en door hun opereren in drukke omgevingen mogelijk te maken.

Er moeten echter nog twee belangrijke beperkingen worden opgelost: de voorgestelde bewegingsplanner berekent open-loop interactiebewuste bewegingsplannen en houdt geen rekening met interactie in closed-loop. Bovendien vertrouwen het voorspellingsmodel en het begeleidingsbeleid uitsluitend op offline leren. Toekomstige werken kunnen onderzoeken hoe rekening kan worden gehouden met interactie in de planningsfase en hoe online datastromen kunnen worden gebruikt om de prestaties van het navigatie-algoritme in de loop van de tijd te verbeteren.

Acknowledgments

From very early in my life, I wanted to be a scientist. I loved to learn new things and make new inventions. Consequently, doing a Ph.D. became an important goal in my life. I dreamed of learning from the best researchers, developing new ideas, and putting them into practice. Now looking back, my Ph.D. was much more than doing research. Not only I grew as a researcher but, more importantly, as a person, and it opened many doors for my future career. Here, I want to thank the people that are part of this journey throughout my Ph.D. and the path that leads to it.

Firstly, I want to thank my daily supervisor and promotor, Dr. Javier Alonso-Mora, for giving me this fantastic opportunity, believing in me, and always supporting and guiding me on the way. I have learned immensely from you: to think critically and to always seek the most novel ideas. I also want to thank my promotor Prof. Robert Babuska for always supporting me during the journey, mainly by allowing me to continue my research in Portugal close to my loved ones during COVID times.

Secondly, I want to thank Prof. Dariu Gavrila, Prof. Caspar Chorus, Prof. Jonathan P. How, Prof. Koen Hindriks, and Prof. Meng Wang for accepting to be my Ph.D. committee members. A mark in my Ph.D. journey was the collaboration with Prof. Jonathan P. How and Dr. Michael Everett. Here, I want to thank them for providing me the opportunity to visit the Aerospace Controls Laboratory (ACL) laboratory at MIT. Unfortunately, the visit had to happen remotely due to the COVID pandemic. Nevertheless, my work with them resulted in what I consider to be the most significant contribution to this thesis. Especially, I want to thank Dr. Michael Everett for all the exciting discussions and for introducing me to the reinforcement learning world.

Thirdly, I am very grateful to my colleagues at TU Delft. Specially, I want to thank Hai Zhu, my Ph.D. brother. Thank you for always being the friend I needed in the best and worst moments of the journey. I have significantly learned from you; to Boaz Floor for all the help in the early stage of my Ph.D. and now, I am happy that our paths have crossed again; to Dr. Michal Čáp for the friendship; to Dr. Laura Ferranti for helping me get my first research paper published and for always being available to revise my articles. I will never forget our initial trials to make the Toyota Prius drive autonomously; to Dr. Carlos Celemin, Giovanni Franzese, Rodrigo Pérez-Dattari for being the amazing friends you were, for all the runs, dinners, and parties that we have shared; to Maximilian Kronmüeller and Max Spahn for introducing me the beautiful landscapes in the Netherlands when cycling; to Luzia Knödler for the help with my dissertation cover; to Lasse Peters for all the lessons on game theory. I have learned a great deal from you; to all my talented master's students: Achin Agarwal, Chadi Salmi, Ewoud Croll, Niels Marcelis, Sant Brinkman and Sukrit Gupta. Thanks for trusting me to be your supervisor. I have learned from you how challenging it is to supervise and guide students but also how rewarding it is.

The path toward my Ph.D. was not straightforward. My first step after finishing my master's degree was an internship at the European Space Agency (ESA). At ESA, I have been luckily introduced to the robotics world. I want to thank Dr. Guillermo Ortega for his support throughout my career and my goal to pursue a Ph.D; to Manuel Sanchez-Gestido for the friendship; to the Sultans: Martin Azkarate, Moisés Navarro, Stella Papadimitriou, Maria Markalain, Alicia Beldina, and Pedro Coelho. Thanks for being the amazing family that you are. Our moments together will always fill my heart with joy; to Micael Miranda and Stéphanie Oliveira for the unconditional friendship and support. You have not only helped me finish this chapter but also brought new light into this world. To the little Mateo I wish you a life full of happiness. After two incredible years in Dutch lands, Germany was next. I was given the opportunity to research robotics at the Fraunhofer institute in Stuttgart. Living in Germany as a foreigner without speaking the language is hard. But, I was blessed with some of the best people in my life. Firstly, I want to thank my brother Marcel Scherrmann. You are the most Portuguese German I have ever met, with a giant heart full of joy. May you continue to rule many cities. To Kevin Bregler, Alexander Pekarovskiy and Richard Bormann. You have enlighted my period at Fraunhofer.

As Jess C. Scott wrote:

> "Friends are the family you choose".

To Irmandade: André Gonçalves, Daniel Castro Neves, Antonia Ćurdo, Diana Correia, Luciano Sousa, Manuel Sampaio, Petra Gouveia, Vasco Seifert, Rita Santos Silva, Pedro Melo, Mafalda Leal Moura, João Leite and Paulo Gouveia, you represent the true meaning of friendship; to Pedro Barbosa and Luciana Costa, you are some of the best persons that I know and I am grateful to have you as friends; to Margarida, you are a great friend and I admire your strength, and your visits are some of the best moments in the Netherlands; to Tiago Sa, you are an example of perseverance; to my Italian brother Simone Fisci who taught me to never stop partying; to Tiago Caetano for being the brother that I wish I had, and for always helping me to stand-up on every fall.

To my family, To my sister Paula, thanks for always taking care of me. You are the person with the biggest heart I know. To Carlitos, thanks for all the help and always receiving me in your house with open arms. To my sister Carla, you will always be my little sister. I am happy that our paths crossed in Delft. To my parents, who supported my studies and me every step of the way. Everything I am and I have achieved, it is because of you. You are my heroes. To Carlinhos and the new family joiners: Maria e Margarida. Your godfather and uncle will always love you. Whenever you need me, I will be there for you.

To Luisa, my love, my wife and my friend. The flapping of a little butterfly's wings brought us together, and I am grateful for all your unconditional support and love. I am the luckiest person to have you in my life.

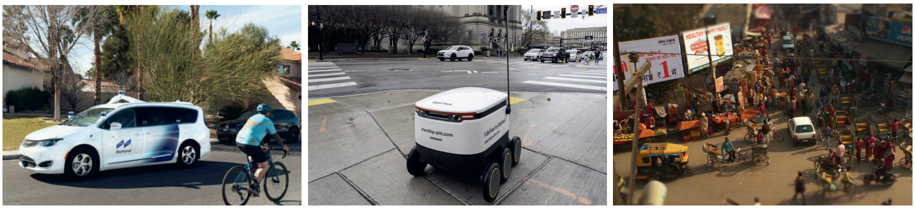
Bruno Brito
Boston, September 2022

1

Introduction

1.1 Motivation

Autonomous robot systems will profoundly impact our society, enabling us to automate transportation and delivery services and assisting us in our households and hospitals [1]. For the first time, robot-taxi companies have announced to begin commercial operations without a safety driver [2] (see Fig. 1.1a). Companies started to test their autonomous last-mile delivery systems on the streets [3] (see Fig. 1.1b). Amazon introduced their first household robot for home monitoring and companionship [4]. Yet, unstructured environments with a large density and number of traffic participants, driving a large variety of vehicles, and exhibiting a wide range of behaviors (see Fig. 1.1c as example), are still challenging. Current autonomous systems are limited to controlled settings, low speed, and clutter-free environments despite all the recent technological advancements. For instance, although autonomous driving companies have been granted the deployment of autonomous vehicles, they can only operate in designated parts of the public roads and during the night and cannot exceed 30 miles per hour [5].



(a) Example of an autonomous riding service (by Motional)

(b) Last-mile delivery system (by Starship Technologies

(c) Dense traffic scenario

Figure 1.1: Illustrative examples of two real-world autonomous robot applications (left and center image) and a challenging scenario for autonomous driving technologies (right figure).

In urban settings, robots must navigate among other decision-making agents (i.e., robots and humans). However, coordinating with other traffic participants is extremely challenging in complex urban environments. While robots can communicate, that is not the case when interacting with humans. Therefore, like humans, robots must be able to infer the other agents' intentions (i.e., prediction), reason about how their actions influence the other agents' (i.e., interaction) and use this information to plan safe and socially compliant motion plans.

This thesis addresses three main problems in autonomous navigation: prediction of pedestrian trajectories, interaction, and guidance. Firstly, while robots can coordinate by sharing their motion plans [6], humans coordinate without explicit communication. Therefore, robots must be capable of predicting what the other road users are likely to do. Secondly, navigating in cluttered environments includes complex interactions among various traffic participants. Hence, algorithms enabling robots to reason about the interactions among multiple traffic participants and planing accordingly are essential for safe and efficient navigation in cluttered scenarios. Finally, in crowded environments, as the one depicted in Figure 1.1c, many different motion plans exist and exploring them all is prohibitive. Therefore, algorithms suggesting a socially intuitive path providing global guidance for the robot are required. This thesis focuses on the motion planning, prediction and decision-making problems.

Over the past decades, there has been an immense effort from the research community and industry to develop planning algorithms to enable safe autonomous navigation in human-populated environments [7, 8]. Nevertheless, most approaches consider well-structured scenarios (e.g., highways) and a limited number of interacting agents. Traditional approaches mostly rely on continuous re-planing [9, 10] or reactive-methods [11–13] to handle the dynamic nature of human environments but do not account for the environment evolution nor for interaction. Additionally, optimization-based methods are widely used in autonomous navigation systems because of their ability to provide safety guarantees. Yet, these methods have scalability issues [14–16] and suffer from *the curse of dimensionality* when accounting for interaction. In contrast, learning-based methods allow to overcome the dimensionality issues by exploiting large amounts of data and offline training. But, these methods do not provide any safety guarantees [17, 18].

In this dissertation, the main research goal is to *develop algorithms that enable safe autonomous navigation in crowded dynamic environments populated with humans and other robots without communication and accounting for interaction.*

1.2 Approach

This thesis investigates learning-based and optimization-based methods to develop planning algorithms accounting for interaction and scaling to cluttered environments. To start, this thesis follows a sequential approach [8], relying on state-of-the-art perception [19] and localization [20] algorithms to perceive the environment, as depicted in Fig. 1.2. Additionally, it is assumed the existence of a high-level mission planning layer providing the mission's goals. Hence, this thesis focus on the motion planning, prediction, and decision-making problems.

1.2.1 Local Motion Planning

The motion planning problem can be in general formulated as follows: given a goal state provided by a mission planner, the current robot's and environments' (e.g., other traffic participants and static obstacles) states, the goal is to plan a local trajectory and the control commands for the robot. The trajectory must respect the robot's kino-dynamic constraints and ensure collision-free motion while progressing towards the goal. For collision avoidance with dynamic obstacles, moving obstacles' are modeled as ellipsoids and the robot's space as circles. This thesis provides a correct bound to approximate the collision region, given by the Minkowsky sum of an ellipse and a circle. Then, this bound is used as a non-linear inequality constraint to ensure safety. Model Predictive Control (MPC) is used as a local motion planning method to enable predictive planning while satisfying the robot's kino-dynamic and collision constraints.

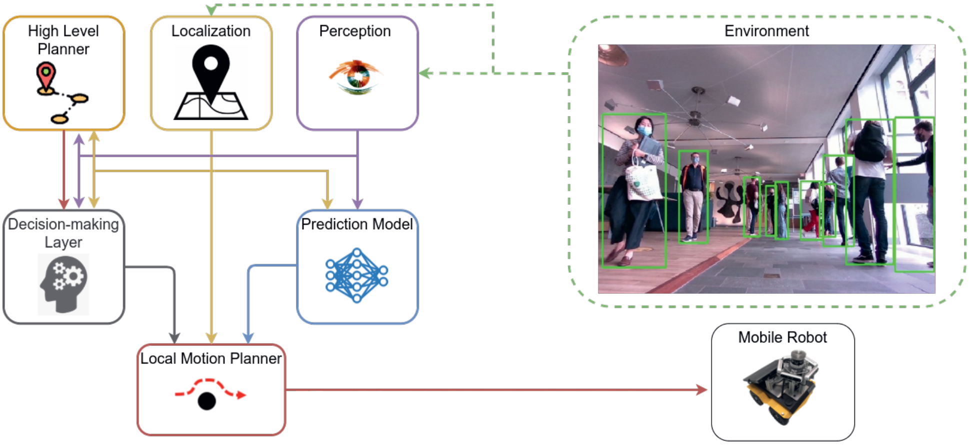


Figure 1.2: Navigation pipeline used in this thesis. This thesis focus on the motion planning, prediction and decision-making modules.

Model Predictive Control

The methods presented for local trajectory optimization in this thesis rely on the MPC framework [21]. MPC enables predictive planning, which is essential to ensure smooth collision avoidance and generate anticipatory behavior. The key idea behind MPC is to compute a sequence of control inputs by optimizing over a finite time horizon, incorporating predictive information about the future environment's states, and planning accordingly. By defining the MPC as an optimization problem, a trajectory minimizing some cost function is computed while satisfying constraints (e.g., collision and model dynamics) if a feasible solution is found. Only the first control input is applied for each time-step, and a new solution is computed considering the current state information.

1.2.2 Trajectory Prediction

Predictive motion planning relies on motion predictions to generate anticipatory behavior. In multi-robot systems, motion predictions can be obtained by having each robot share their future planned trajectories via communication [22]. However, when navigating among humans, such a communication channel is not available. Therefore, autonomous robots, such as service robots or autonomous cars, must reason about the intentions of the other agents (e.g., pedestrians, other robots, etc.) and forecast their motions accurately.

To build an inference model of other agents' motion is extremely difficult due to its uncertainty and multimodality (i.e., under the same circumstances, an agent may follow multiple and different paths). The uncertainty is caused by the partial observation of the other agents' states and their stochastic dynamics. The multimodality is due to the interaction effects between the agents, the static environment and the non-convexity of the problem. Hence, most state-of-the-art planners employ simple prediction models (e.g., a constant velocity model [23]) not accounting for interaction and multi-modality. However, predictive planning methods' performance highly depends on the prediction accuracy and the ability of prediction models to reason about interaction.

Recent advances in the field of parallel computing and automatic differential tools [24] enabled machine learning methods to learn prediction models leveraging high-dimensional information and large amounts of data. This thesis employs deep learning to create a model providing high-fidelity multimodal predictions. Chapter 4 presents a local motion planner building on MPC for autonomous robot navigation in unstructured dynamics environments.

Deep Learning

In general, the trajectory prediction problem can be formulated as a supervised learning problem: given a dataset containing a set of input features (e.g., agent velocity and position) and the associated true label or ground-truth, the model parameters θ and a loss function (e.g., mean square error (MSE), log-likelihood, etc.,.), one can compute the optimal model parameters θ^* using gradient descent. As a model, most recent algorithms rely on DNNs because of their ability to incorporate high-dimensional information and infinity approximation capacity [25]. When combined with variational learning [26], or normalizing flows [27], NNs can also be used to learn approximations of the model's probability density function from data, enabling uncertainty and multi-modality into the model predictions. Chapter 5 provides a detailed overview on the applicability of variational learning in the context of trajectory prediction.

1.2.3 Learning for Global Guidance

Safety and efficiency are two essential features for the successful deployment of autonomous robots in human environments. When it comes to safety, optimization-based methods can compute plans respecting important constraints (e.g., kinematics and collision avoidance) [7]. However, to enable online optimization these approaches rely on several assumptions and simplifications that become impracticable in dense scenarios populated with humans.

Firstly, optimization-based algorithms (e.g., MPC) often employ a limited planning horizon to limit the problem's complexity and enable real-time computation. Consequently, the solutions computed may be locally optimal and may lead to collisions or deadlocks events in the long term. Secondly, optimization-based method's performance strongly depends on high-level decision variables typically defined as constant hyper-parameter values (e.g., cost weights, reference values, etc.), which are a hurdle to tune. Moreover, high-level decision variables significantly influence the robot's behavior and may hurt the problem's feasibility.

This thesis explores machine learning methods, specifically Reinforcement Learning (RL), to provide global guidance on high-level decision variables enabling local planning methods to scale to cluttered environments and compute interaction-aware motion plans.

Reinforcement Learning

RL enables learning a decision-making policy offline by trial-and-error by interacting with a simulation environment. Typically, the goal in RL is to learn a policy that maximizes rewards over an infinite time horizon. The key advantage of using RL is that it allows moving the *curse-of-dimensionality* burden in online planning to an offline training phase. The learned policy allows taking decisions accounting for the short and long-term impact. Hence, RL algorithms are widely used to learn decision-making and control policies [28]. When using NNs as the policy's model (i.e., Deep Reinforcement Learning (DRL)), RL can account for high-dimensional information such as RGB-D images or LiDAR data.

Chapter 6 and Chapter 7 propose two novel approaches employing DRL to learn global guidance policies with applications to dense traffic scenarios and autonomous navigation among humans and other robots.

1.3 Contributions and Outline

The goal of this thesis is to provide algorithmic innovations to enable safe and efficient robot navigation among humans, as discussed in Sec. 1.1. Hence, this thesis makes the following scientific contributions:

(1) **A Local Model Predictive Contouring Control (LMPCC)** approach for local motion planning in unstructured dynamic environments. By computing a set of convex regions representing the free space from a static map and modeling moving obstacles (e.g., pedestrians) as ellipsoids, the method allows computing motion plans in real-time. This method enabled an autonomous mobile robot to safely navigate in indoor environments populated with humans and an autonomous vehicle to perform a collision avoidance maneuver to avoid a pedestrian crossing the road.

(2) **An interaction-aware variational recurrent neural network (Social-VRNN)** model for one-shot multi-modal trajectory prediction. The proposed variational Bayesian approach enables faster training convergence than GAN-based approaches. Moreover, employing a time-dependent prior over the latent space enables the proposed model to achieve state-of-the-art performance and generate diverse trajectories with a single network query.

(3) **A framework enhancing state-of-the-art online optimization-based planners with a learned global guidance policy** applied to two different navigation problems:

    (a) **A goal-oriented Model Predictive Control method (GO-MPC)** for navigation among interacting agents. The GO-MPC utilizes a learned global guidance policy (recommended subgoal) in the cost function, minimizing a long-term cost-to-go and accounting for interaction. By relying on an MPC as a local planner, the GO-MPC ensures dynamic feasibility and collision avoidance when a feasible solution is found. Moreover, jointly training the RL policy with an optimization-based planner allows using the proposed method directly on real hardware, reducing the sim to real gap.

(b) **An Interactive Model Predictive Controller (IntMPC)** for navigation in dense traffic environments. By combining DRL to learn an interaction-aware policy providing information on high-level decision variables (e.g., velocity reference) directly in the local planner's cost function, the proposed approach enables interactive behavior mandatory to navigate in dense traffic scenarios successfully.

The proposed algorithms of this thesis have been extensively evaluated and validated with a mobile robot platform and a self-driving vehicle, see Appendix A and Appendix B for more details on the platforms, respectively.

Figure 1.3 presents the overall thesis structure. Initially, Chapter 2 presents the fundamentals of MPC, supervised learning, and reinforcement learning to support the introduction of the following chapters. Chapter 3 reviews the state of the art of motion planning methods in dynamic environments, followed by trajectory prediction models, learning global guidance policies and interaction-aware motion planning. Chapter 4 presents the local model predictive contouring control (LMPCC) method for autonomous navigation in unstructured dynamic environments. Chapter 5 introduces the Social-VRNN model for single-shot multi-modal trajectory predictions. Chapter 6 presents the Goal-oriented Model predictive controller for navigation among humans and other robots. Chapter 7 describes the Interactive Model Predictive Controller for interaction-aware motion planning in dense traffic scenarios. Finally, Chapter 8 draws the conclusions of this thesis and presents possible future research directions.

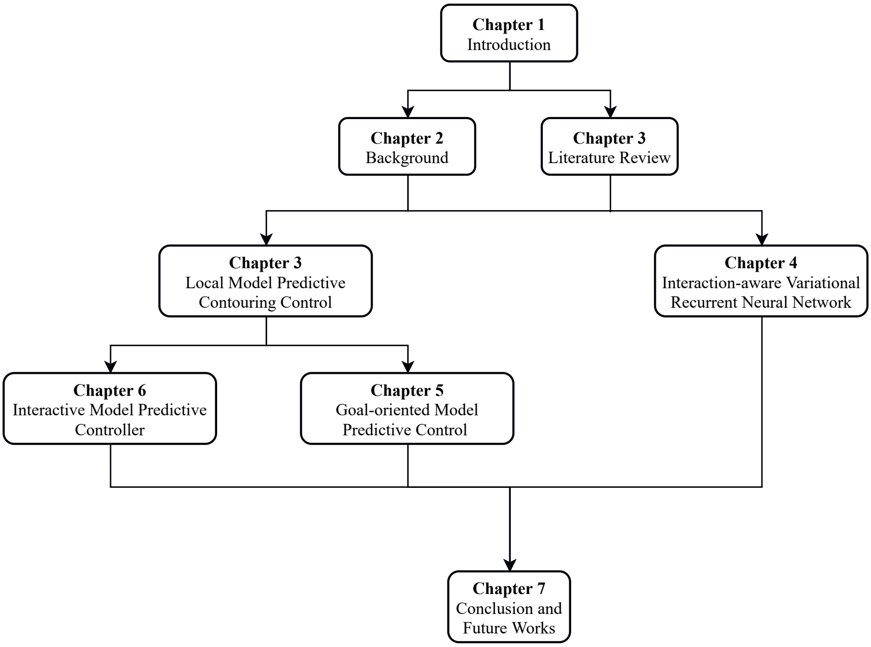

Chapter 1
Introduction
Chapter 2
Background
Chapter 3
Literature Review
Chapter 3
Local Model Predictive Contouring Control
Chapter 4
Interaction-aware Variational Recurrent Neural Network
Chapter 6
Interactive Model Predictive Controller
Chapter 5
Goal-oriented Model Predictive Control
Chapter 7
Conclusion and Future Works

Figure 1.3: Thesis' structure: Chapter 1 introduces and motivates the research presented on this thesis. Next, the fundamentals of the methods proposed in this thesis are presented (Chapter 2), followed by the related works (Chapter 3). Chapter 4 presents the first contribution of this thesis: a local motion planner serving as the basis of the contributions presented in Chapter 7 and Chapter 6. At the same level, Chapter 5 introduces the proposed multi-modal prediction model. Finally, Chapter 8 presents the conclusions and proposed future research directions.

1.4 Notation

This section describes the general and common notation used throughout this thesis.

The scalars are denoted by italic lowercase letters, x, vectors by bold lowercase, $\mathbf{x}$, matrices by plain uppercase, M, and sets by calligraphic uppercase, $\mathcal{S}$. To denote the transpose of a vector or matrix the superscript $\mathbf{x}^T$ or M^T is used, respectively. $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}$, $\|\mathbf{x}\|^2 = \mathbf{x}^T\mathbf{x}$, and $\|\mathbf{x}\|_Q^2 = \mathbf{x}^T Q\mathbf{x}$ denote the Euclidean norm, squared Euclidean norm, and weighted squared Euclidean norm of $\mathbf{x}$, respectively. $\dot{\mathbf{x}}$ denotes the derivative of $\mathbf{x}$ with respect to time t.

2

Background

This chapter briefly introduces the fundamental methods on which the proposed approaches in this dissertation are built. Firstly, we introduce the Model Predictive Control framework, and then, we provide a gentle introduction to supervised learning and reinforcement learning. Overall, this chapter presents essential preliminaries to the remainder of this thesis.

2.1 Model Predictive Control

Optimization methods (e.g., dynamic programming) are widely used for planning and control because of their ability to solve problems with equality and inequality constraints [29]. The basis for optimization problems is a dynamic model describing the state evolution of the system:

$$x_{k+1} = f(x_k, u_k) \tag{2.1}$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ are the state and control input vectors, respectively.

Here, we will only consider discrete time dynamics for sake of clarity and simplicity. To handle continuous dynamics, there are several methods to discretize a continuous time system [30]. In addition to the dynamics model, the optimization problem objectives (e.g., follow route, fuel consumption or travel time) are defined as a cost function:

$$J(x,u) = \sum_{k=0}^{T-1} J_k(x_k, u_k) + J_T(x_T, u_T) \tag{2.2}$$

where J_k and J_T are the stage and end cost function, respectively.

Generally, an optimization problem is formulated as

$$\begin{aligned}
u^* \quad &= \arg\min_{u} \sum_{k=0}^{T-1} J_k(x_k, u_k) + J_T(x_T, u_T) && \text{(2.3a)}\\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k) && \text{(2.3b)}\\
& g(x_k, u_k) \leq 0 && \text{(2.3c)}\\
& x_k \in \mathcal{X}, \quad u_k \in \mathcal{U} \quad \forall k \geq 0 && \text{(2.3d)}\\
& x_0 = x_{\text{init}} && \text{(2.3e)}
\end{aligned}$$

where x_0 represents the initial conditions, g the inequality constraint function and $\mathcal{X}$ and $\mathcal{U}$ are the sets of admissible states and control inputs, respectively. However, the problem becomes intractable for long planning horizons.

Model Predictive Control addresses the last issue by considering a limited planning horizon, repeatedly solving a finite time optimal control problem in a receding horizon fashion. At each time step k, starting at the current state (Equation (2.3e)), an open-loop optimal control problem is solved over a finite horizon T to obtain a locally optimal sequence of control commands $u^* = [u_0, \ldots, u_{T-1}]$ (see Figure 2.1). Then, only the first

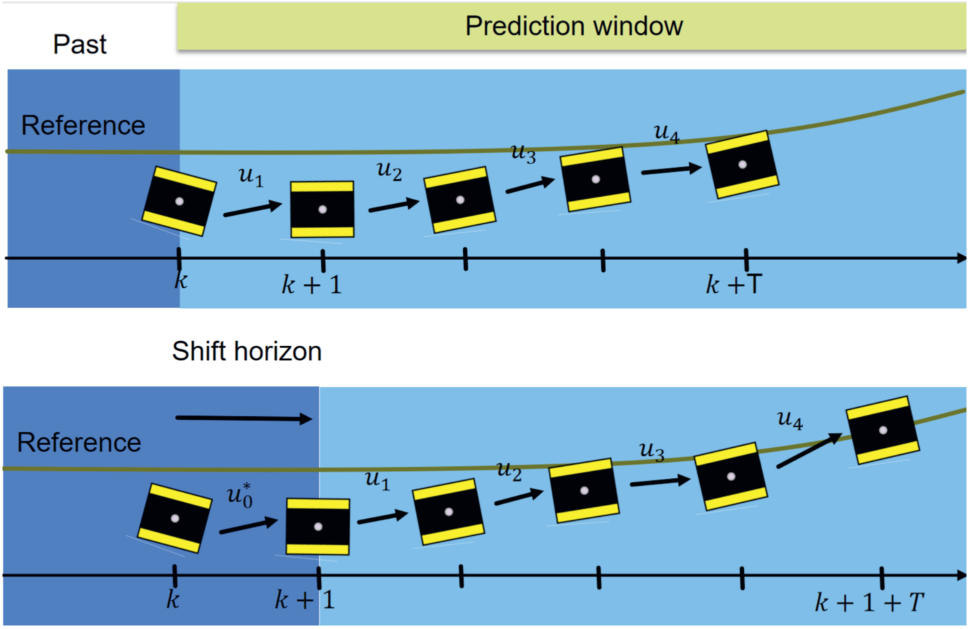


Figure 2.1: Receding horizon strategy. At time step k, MPC computes locally optimal sequence of control commands $u_{0:T}$ enabling the robot to minimize the distance to the reference path depicted in green. Only the first control command u_0^* is applied and at the next time step $k+1$ the MPC computes a locally optimal sequence of control commands given the new observed information by the robot.

control command is applied and at the next time step $k+1$ a new optimal control problem based on new state information is solved over a shifted horizon (see bottom Figure 2.1). Thus, MPC allows to compute online a locally optimal solution to the infinite horizon controller problem.

2.2 Supervised Learning

With the recent advancements of automatic differentiation tools and parallel computing, learning high-dimensional models (i.e., deep learning) from large amounts of data is now possible. This has triggered the research on supervised learning applied to a broad spectrum of applications (e.g., high-fidelity prediction models, high-performance visual detectors, etc,.) [19, 31]. For instance, end-to-end driving policies [32] or high-fidelity pedestrians prediction models [33] have been learned from offline datasets. This section provides a gentle introduction on supervised learning and how to learn deterministic and probabilistic models from data [34, 35].

Consider the task of learning a function f_θ with parameters θ mapping an input vector $\mathbf{x}$ to an output vector $\mathbf{y}$

$$\hat{\mathbf{y}} = f_\theta(\mathbf{x}) \tag{2.4}$$

where $\hat{\mathbf{y}}$ is the model's output vector which can be defined as a sequence to learn a prediction model or a control vector to learn a policy. In a supervised learning setting, there is a dataset $\mathcal{D}$ containing N tuples of input and output vectors, $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots (\mathbf{x}_N, \mathbf{y}_N)\}$. These tuples typically correspond to expert labels, $\mathbf{y}$, given the observation $\mathbf{x}$. The goal is to find the model's parameters θ^* minimizing a loss function

$$\theta^* = \arg\min_\theta L(\theta) \tag{2.5}$$

where L is the loss function used. When learning a deterministic model, L is typically defined as the mean-squared error between the ground-truth vector $\mathbf{y}$ and the models output $\hat{\mathbf{y}}$, the norm loss or the Huber-loss [36]. When learning probabilistic models, the *negative log-likelihood* is normally used resembling maximum likelihood estimation (MLE). To avoid models to overfit the dataset and improve generalization of the learned model extra loss terms can be added to the loss function (e.g., L2 and L1 regularization) [34]. The necessary conditions for θ^* to be a minimum are $\frac{\partial L}{\partial \theta}(\theta^*) = 0$. However, when using non-linear and high-dimensional models such as neural networks there is no close-form solution and θ^* can only be found via iterative numerical optimization (e.g., gradient descent algorithms). Therefore, we start from a initial set of model's parameters θ_0 and for each training step we update the model's parameters in the direction that minimizes the loss function (i.e., negative gradient direction). Algorithm 1 presents the overall algorithm.

Algorithm 1 Gradient Descent

```
Given: initial model parameters $\theta_0$, step size $\alpha$, the number of training steps $n_{\text{steps}}$
    and batch size $n_{\text{batch}}$
for $k = 0, 1, 2, \ldots n_{\text{steps}}$ do
    Randomly sample $n_{\text{batch}}$ examples from the dataset
    Update the model's parameters:
        $\theta_{k+1} = \theta_k + \alpha \frac{1}{n_{\text{batch}}} \sum_{i=1}^{n_{\text{batch}}} \nabla_\theta L(\mathbf{x}_i, \mathbf{y}_i)$
end for
```

Techniques that use the whole dataset at once are called batch methods. For large datasets batch methods are intractable. Therefore, *mini*-batch methods split the training dataset into small batches to compute the update direction vector. Methods that make an update to the parameters vector based on one data point at a time, $n_{\text{batch}} = 1$, are known as sequential gradient descent or stochastic gradient descent and are typically used for online learning.

2.2.1 Variational Bayes

Let us consider the problem of learning a probability distribution of a random variable $\mathbf{x}$ from a dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}$ generated by some random process involving an unobserved continuous random variable $\mathbf{z}$. Moreover, $\mathbf{z}$ is generated from some prior distribution $q_\phi(\mathbf{z})$. Stochastic Gradient Variational Bayes (SGVB) [26, 37] tackles the intractability problem of the integral of the marginal likelihood $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$ and the true posterior density $p_\theta(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})/p_\theta(\mathbf{x})$. Additionally, it addresses the problem of learning from large datasets where batch optimization is too expensive. The key idea is to learn an approximate model for the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and for the marginal distribution $p_\theta(\mathbf{x}|\mathbf{z})$, and optimizing an evidence lower bound (ELBO):

$$\mathcal{L}_{\text{ELBO}}(\theta, \phi; \mathbf{x}_i) = KL(q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z})) - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}_i|\mathbf{z})] \tag{2.6}$$

where KL denotes the the Kullback-Leibler divergence of p from q. However, we still need to sample from the approximate posterior distribution $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ during the training. To enable learning both models end-to-end through back-propagation, we can obtain a continuous differentiable sampler by applying the reparametrization trick [26]:

$$\mathbf{z} = \mu + \sigma * \epsilon \tag{2.7a}$$

$$\mathbf{z} \sim \mathcal{N}(\mu, \sigma) \tag{2.7b}$$

where ϵ is an auxiliary noise variable $\epsilon \sim \mathcal{N}(0, 1)$. Here, we assume that the approximate posterior distribution follows a Gaussian distribution (Equation (2.7b)).

2.3 Reinforcement Learning

Markov Decision Processes (MDPs) are widely used to model sequential decision making problems. An MDP is defined by a set of states $\mathcal{S}$ and actions $\mathcal{A}$, a transition function $P(s, s'|a)$ describing the probability of transitioning to state s' from state s by taking action a, and a reward function $r(s, a, s')$ providing quantitative feedback on the action taken. A trajectory generated by sampling actions according to the policy π is defined as a sequence of states and actions, $\tau \sim \pi = (s_0, a_0, \ldots, s_T, a_T)$. MDPs rely on two main assumptions [38]:

1. The problems are formulated as a sequence of independent decisions.
2. The state at time t depends only on the events at time $t-1$.

The optimality of a sequence of decisions (i.e., actions) is measured by return (i.e., discounted sum of future rewards):

$$R = \sum_{k=0}^{K} \gamma^k r_k \tag{2.8}$$

where R is the return, r_k the immediate reward at time step k, $K \in [0, \infty)$ is the episode length and $\gamma \in (0,1]$ is a discount factor balancing the importance of short and long term rewards. In the reinforcement learning framework, an agent learns by interacting with the environment and using the sequence of states, actions and rewards to improve its decision making policy [39], as depicted in Figure 2.2.

In general, RL can be divide in two main phases: experience collection and policy training.

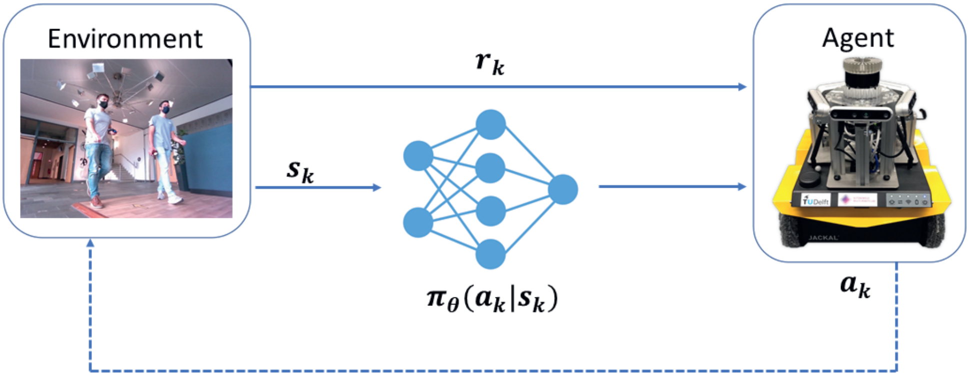


Figure 2.2: RL framework: at each time-step k, the robot first observes its state s_k, then takes action a_t, leading to the immediate reward r_k and next state $s_{k+1} = P(s_k, a_k)$, under the transition model P.

The goal of reinforcement learning algorithms is to find the optimal policy's parameters θ maximizing the return for a initial distribution of states $\mathcal{S}_0$:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \quad \mathbb{E}_{s_0 \sim \mathcal{S}_0, a \sim \pi_\theta} R(s_0) \tag{2.9}$$

2.3.1 Proximal Policy Optimization

Policy gradients aim to maximize the expected total reward by estimating the gradient of the expected total reward and optimize the policy parameters by stochastic gradient ascent. Commonly, policy gradients estimates have the following form:

$$\nabla_\theta L = \mathbb{E}\{\sum_{k=0}^{\infty} \hat{A}_k \nabla_\theta \log \pi_\theta(a_k|s_k) \tag{2.10}$$

where π_θ is a stochastic policy, θ policy's parameters, $\mathbb{E}$ denotes the expectation and $\hat{A}_k$ is an estimate of the advantage function at timestep k. An n-step estimate of the advantage function is computed as follows:

$$\hat{A}^n(s_k, a_k) = \sum_{l=0}^{n-1} \gamma^l r_{k+l} + \gamma^n \hat{V}(s_{k+n}) - \hat{V}(s_k) \tag{2.11}$$

Bootstrapping methods (low n) lead to biased estimates and so, policy updates. In contrast, Monte-Carlo methods (large n) solve the bias issue but suffer from high variance. Generalized Advantage Estimation (GAE) [40] proposes an optimize solution for the bias-variance trade-off by computing an exponentially weighted average of the n-step advantage estimator:

$$\hat{A}_k^{GAE}(\lambda) = (1-\lambda) \sum_{l=0}^{N} \lambda^{n-1} \hat{A}^n \tag{2.12}$$

Nevertheless, getting good results via policy gradient methods is challenging because they are highly sensitive to the choice of step size. Too small step size leads to slow learning progress, and too large may lead to catastrophic drops in performance resulting in poor sample efficiency and requiring a large number of steps to learn simple tasks.

Proximal Policy Optimization (PPO) [41] solves the latter by proposing to constraint the differences between the policy distribution before and after the optimization step. Thus, avoiding catastrophic changes in the policy parameters by clipping gradients of the surrogate objective:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{\text{old}}}}[\min(ratio(\theta)\hat{A}^{GAE}, clip(ratio(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^{GAE})] \tag{2.13}$$

where ϵ is a hyperparameter limiting the policies changes and $ratio(\theta) = \frac{\log \pi_\theta}{\log \pi_{\theta_{\text{old}}}}$ is the ratio of logarithmic probability between the new and old policies.The probability ratio term in the PPO objective employs the idea of importance sampling, allowing to evaluate of the new policy with samples collected from the old policy and thus, improving sample efficiency [42].

In addition to the surrogate objective, PPO optimizes two extra objectives: a regression loss to learn a value function enabling parameter sharing [43] between the policy and value function and the policy's entropy to motivate exploration [44].

$$L^{\text{PPO}} = L^{\text{CLIP}} + L^{\text{v}} + L^{entropy} \tag{2.14}$$

Algorithm 2 presents the PPO pseudocode. Overall, PPO is easy to implement, sample efficient and easy to tune.

Algorithm 2 PPO-Clip [41]

```
Input: initial policy parameters $\theta_0$, initial value function parameters $\phi_0$
for k=0,1,2, ... do
    Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
    Compute returns estimates $\hat{R}_t$ .
    Compute advantage estimates, $\hat{A}_t^{GAE}$ (using GAE [40]) based on the current value function $V_{\phi_k}$.
    Update the policy by maximizing the PPO-Clip objective:
        $\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k|T} \sum_{\tau\in\mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t,a_t), \quad g\left(\epsilon, A^{\pi_{\theta_k}}(s_t,a_t)\right)\right)$
    typically via stochastic gradient ascent with Adam [45].
    Fit value function by regression on mean-squared error:
        $\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k|T} \sum_{\tau\in\mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t\right)^2,$
    typically via some gradient descent algorithm.
end for
```

2.3.2 Soft Actor-Critic

Soft Actor-Critic (SAC) [46] learns a stochastic policy with entropy regularization, and explores in an on-policy way. In an entropy-regularized reinforcement learning setting, the goal is to find a policy's parameters maximizing the expected return and the policy's entropy:

$$\pi^* = \arg\max_\theta \underset{\tau\sim\pi_\theta}{\mathrm{E}} \left[\sum_{t=0}^{\infty} \gamma^t \left(R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) + \alpha H(\pi(\cdot \mid \mathbf{s}_t))\right)\right] \tag{2.15}$$

where H is the entropy function and $\alpha > 0$ is the exploration-exploitation trade-off coefficient.

There are different variants of the SAC method. Here, we introduce the SAC variant learning two Q-functions Q_{ϕ_1}, Q_{ϕ_2}. Moreover, to enhance training stability SAC employs two target networks $Q_{\phi_{\text{targ},1}}, Q_{\phi_{\text{targ},2}}$.

Algorithm 3 Soft Actor-Critic [46]

```
1:  1: Input: initial policy parameters $\theta_0$, Q-function parameters $\phi_1, \phi_2$ and empty replay buffer $\mathcal{D}$
2:  Set target network parameters to initial parameters $\phi_{\text{targ},1} \longleftarrow \phi_1, \phi_{\text{targ},2} \longleftarrow \phi_2$
3:  **repeat**
4:  Observe state $\mathbf{s}$ and take action $\mathbf{a} \sim \pi_\theta(\mathbf{s})$
5:  Observe next state $\mathbf{s}'$, reward $r$ and terminal signal $d$
6:  Store $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}', terminal)$ in replay buffer $\mathcal{D}$
7:  **if** terminal **then**
8:      Reset environment
9:  **end if**
10: **if** it's time to update **then**
11:     **for** j in range($n_{\text{updates}}$) **do**
12:         Randomly sample a batch of transitions $B = \{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}', \text{terminal})\}$ from $\mathcal{D}$
13:         Compute target values:
                $$y\left(r, \mathbf{s}', \text{terminal}\right) = r + \gamma(1 - \text{terminal})\left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}\left(s', \tilde{\mathbf{a}}'\right) - \alpha \log \pi_\theta\left(\tilde{\mathbf{a}} \mid \mathbf{s}'\right)\right) \quad (2.16)$$
                **s.t.** $\tilde{\mathbf{a}}' \sim \pi_\theta\left(\cdot \mid \mathbf{s}'\right)$
14:         Update Q-functions by one step of gradient descent using
                $$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}', \text{terminal}) \in B} \left(Q_{\phi_i}(\mathbf{s}, \mathbf{a}) - y\left(r, \mathbf{s}', d\right)\right)^2 \quad \text{for } i = 1, 2 \quad (2.17)$$
15:         Update policy by one step gradient ascent using
                $$\nabla_\theta \frac{1}{|B|} \sum_{\mathbf{s} \in B} \left(\min_{i=1,2} Q_{\phi_i}\left(\mathbf{s}, \tilde{\mathbf{a}}_\theta(\mathbf{s})\right) - \alpha \log \pi_\theta\left(\tilde{\mathbf{a}}_\theta(\mathbf{s}) \mid \mathbf{s}\right)\right) \quad (2.18)$$
            where $\tilde{\mathbf{a}}(\mathbf{s})$ is a sample from $\pi_\theta(\cdot|\mathbf{s})$ which is differentiable wrt $\theta$ via the reparametrization trick
16:         Update target networks using Polyak averaging
                $$\phi_{\text{targ},i} \longleftarrow \rho \phi_{\text{targ},i} + (1 - \rho)\phi_i \quad \text{for } i = 1, 2 \quad (2.19)$$
17:     **end for**
18: **end if**
19: **until** convergence
```

Algorithm 3 presents the SAC algorithm composing two phases: an exploration phase (lines 4-9) and policy and Q-functions training phase (lines 10-17). During the exploration phase, the agent uses the current policy to explore the environment and records the sequence of state, action and reward tuples into a buffer replay. During the training phase, first, a batch of transitions is sampled from the replay buffer (line 12). Then, the target values

y are computed using the smaller Q-value for the target helping to fend off overestimation in the Q-function (Equation (2.16)). The Q-functions are updated by giving a gradient step to minimize the mean square Bellman error (Equation (2.17)). The policy is updated by giving a gradient ascent step to maximize the expected return and entropy (Equation (2.18)). Finally, the target Q-functions are updated using Polyak averaging between the main and target parameters (Equation (2.19)).

3

Literature Review

In this chapter, we summarize related works connected to the three main research problems of this thesis: motion planning in dynamic environments, trajectory prediction and learning global guidance policies. We start with reviewing the state of the art of motion planning methods, followed by a discussion on multi-robot local motion planning.

3.1 Motion Planning in Dynamic Environments

Over the past decades, there has been a tremendous amount of works proposing motion planning algorithms for autonomous navigation in dynamic environments, ranging from sampling-based to learning-based methods. Although some methods can be applied to both mobile robots and autonomous vehicles, there are some critical differences in the assumptions and problem settings. Hence, in Section 3.1.1 we review the literature of motion planning methods applied to mobile robots and in Section 3.1.2 to autonomous vehicles.

3.1.1 Mobile Robots

Collision avoidance in static and dynamic environments can be achieved via reactive methods, such as time-varying artificial potential fields [47], the dynamic window [11], social forces [48] and velocity obstacles [49]. Although these approaches work well in low speed scenarios, or scenarios of low complexity, they produce highly reactive behaviours. More complex and predictive behaviour can be achieved by employing a motion planner. For instance, model predictive control (MPC) [21, 50] allows to obtain smooth collision-free trajectories that optimize a desired performance index, incorporate the physical constraints of the robot and the predicted behavior of the obstacles.

Due to the complexity of the motion planning problem, path planning and path following were usually considered as two separate problems. Many applications of MPC for path-following control are found in the literature, e.g., [51, 52]. These methods assume the availability of a collision-free smooth path to follow.

Several approaches exist for integrated path following and control for dynamic environments. These include: employing a set of motion primitives and optimizing the control commands to execute them [53] and offline computation of tracking error bounds via reachability analysis that ensures a safety region around the robot for online planning [54, 55]. These approaches, however, do not allow to incorporate the predicted intentions of the dynamic obstacles and consequently, can lead to reactive behaviors. To overcome the latter issue, [56] proposed the model predictive contouring control (MPCC) that allows to explicitly penalize the deviation from the path (in terms of contouring and lateral errors) and include additional constraints. Later, [57] designed a MPC for path following within a stable handling envelope and an environmental envelope. While [57] is tailored to automotive applications without dynamic obstacles, MPCC has been employed to handle static and dynamic obstacles in structured driving scenarios [58]. There, static collision constraints were formulated as limits on the reference path and thus limited to on-the-road driving scenarios. The previous approaches do not account for the interaction effects between the agents and may fail in crowded scenarios, a problem known as the freezing robot problem

(FRP) [59]. Interaction Gaussian Processes (IGP) [60] can be used to model each individual's path. The interactions are modeled with a nonlinear potential function. The resulting distribution, however, is intractable, and sampling processes are required to approximate a solution, which requires high computational power and is only real-time for a limited number of agents.

Learning-based approaches address this issue by learning the collision-avoidance strategy directly from offline simulation data [61], or the complex interaction model from raw sensor readings [62]. Yet, both methods learn a reactive collision avoidance policy and do not account for the kinodynamic constraints of the robot.

3.1.2 Autonomous Driving

Recently, increasing attention has been dedicated to VRUs safety (e.g., [63–68]). In [63], a joint team from Daimler and Karlsruhe Institute of Technology drove an autonomous car on the Bertha Benz Memorial Route, where they had to deal with VRUs. Their planner is divided into a behavior generation and a trajectory planning. The behavior generation decides how to interact with static and dynamic obstacles using a state machine. The trajectory planner computes the desired path (without taking into account the dynamics of the vehicle) and sends it to a path-follower low-level controller. When planning the trajectory decisions concerning the obstacles have already been made.

Commercial Autonomous Emergency Braking (AEB) systems are able to avoid collisions with detected VRUs as long as there is a sufficiently large distance between the vehicle and the VRU. In [64], the authors presented a pedestrian AEB analytical model to calculate the certainty of finding a detected pedestrian in the collision zone, by analysing the pedestrian lateral behavior. Their model can help verify existing AEB systems and design new AEB systems.

If the distance to perform an emergency brake is too small, evasive steering maneuvers are required. Research on evasive steering maneuvers for active pedestrian safety is extremely active. In [65], the authors provide a driver-assistant design to decide whether to brake or evade the crossing pedestrian based on the information provided by the perception module. A situation analysis module automatically evaluate the criticality of the current driving scenario. Then, a decision module decides whether to warn the driver or to trigger the appropriate maneuvers for collision avoidance and mitigation using dedicated controllers. In [66], the authors provide an overview of evasive steering techniques discussing the potential of evasive steering vs. braking. In addition, they also detail the design of the Daimler automatic evasion driver-assistance system for pedestrian protection. Similar to [65], their system also relies on a situation analysis module and a decision module that can take over control of the car to trigger an emergency maneuver. In [67], the authors propose an autonomous lane-keeping evasive maneuver that relies on the road infrastructure (cameras placed at specific hazardous locations). Their method can be used to take over control of the car to avoid collisions with a pedestrian when braking is no longer possible. In [68], the authors present a driver assistance system to help the driver initiate an evasive maneuver with pedestrians. The system is able to take decisions by taking into account upcoming traffic.

Sampling-based Methods

RRTs [69] can quickly search in high-dimensional spaces that have both algebraic restrictions arising from obstacles, and differential constraints arising from nonholonomic and dynamic constraints. Hence, it makes the algorithm extremely useful for real-time applications. The algorithm incrementally builds a tree of paths by connecting randomly sampled configurations using an exact [70] or approximate [69] steer function based on vehicle kinematics or dynamics. The advantage of approximate steer methods is that they avoid the need for solving a Boundary Value Problem (BVP) for each sample, which can be a computationally expensive operation for differentially constrained systems [71]. Although RRTs have many benefits concerning traditional combinatorial path planning methods, the algorithm often converges to a sub-optimal solution [70]. This sub-optimality may express itself in the path quality by e.g., producing a meandering path.

Early variants of the algorithm enhance the solution quality by adding an optimization heuristic during nearest neighbor selection [72–74], or prune the path after a solution is found [75]. However, this does not guarantee asymptotic optimality. Therefore, the RRT* introduces a rewiring operation that can guarantee asymptotic convergence towards the optimal solution [70, 76, 77]. Due to the number of BVP that must be solved during rewiring, the real-time implementation may be restricted to the use of exact steer functions only (e.g., [78]).

To avoid the need for solving a BVP, [72, 79] introduced the Closed Loop Rapidly-exploring Random Tree (CL-RRT). This variant samples in the controller's input space (e.g., a path and reference velocity) instead of the vehicle inputs (e.g., steer angle and velocity). The controller inputs are used for simulating a closed-loop trajectory of the vehicle controlled by a lateral and a longitudinal controller.

3.2 Trajectory Prediction

Recently there has been an increased focus on improving prediction methods to enable safe autonomous navigation in urban environments. Hence, the literature tackling the prediction problem for pedestrians and other human-driven vehicles is vast, ranging from model-based (see Section 3.2.1) to data-driven (see Section 3.2.2) methods.

3.2.1 Traditional Approaches

Early works on human motion prediction are typically model-based. In [80], a model of human-human interactions was proposed by simulating attractive and repulsive physical forces denominated as "social forces". To account for human-robot interaction, a Bayesian model based on agent-based velocity space was proposed in [81]. However, these approaches do not capture the multi-hypothesis behavior of the human motion. To accomplish that, [82] proposed a path prediction model based on Gaussian Processes, known as interactive Gaussian Processes (IGP). This was done by modeling each individual's path with a Gaussian Process. The main drawbacks of this approach are the usage of hand-crafted functions to model interaction, limiting their ability to learn beyond the perceptible effects, and is computationally expensive.

3.2.2 Deep Learning

Recently, Recurrent Neural Networks (RNNs) have been employed in trajectory prediction problems [83]. Building on RNNs, a hierarchical architecture was proposed in [84] and [85], which incorporated information about the surrounding environment and other agents, and performed better than previous models. Despite the high prediction accuracy demonstrated by these models, they are only able to predict the average behavior of the pedestrians.

In contrast, Social LSTM [86] models the prediction state as a Bivariate Gaussian and thus, uncertainty can be incorporated. Moreover, interaction is modeled by changing the hidden state of each agent network according to the distance between the agents, a mechanism know as "Social pooling". Several approaches extended the latter either by incorporating other sources of information or proposing updates in the model architecture improving the performance of the model. For instance, head pose information from the other agents was incorporated in [87] resulting in a significant increase of the prediction accuracy. Context information from visual images was used to encode both human-human and human-space interactions [88]. Social pooling has been extended to generate collision-free predictions [89] and to preserve spatial information by employing Grid LSTMs [90].

However, previous approaches did not consider the inherent multi-modal nature of human motions. In [91], a generative model based on Generative Adversarial Networks (GANs) was developed to generate multi-modal predictions by randomly sampling from the latent space. This approach was extended with two attention mechanisms to incorporate information from scene context and social interactions [92]. However, GANs are very susceptible to mode collapsing causing these models to generate very similar trajectories. To avoid mode collapse, a recently improved Info-GAN for multi-modal trajectory prediction was proposed [93]. Besides, [94] proposed a different training strategy to overcome the latter issue and improve trajectory prediction diversity. To account for the environment constraints, [95] proposed to include scene context information provided by a top-view camera of the scene. However, such information is not available in a real autonomous navigation scenario. Moreover, to improve social interaction modelling, Graph Neural Networks have been used in [96, 97]. Nonetheless, GANs are very difficult to train and typically require a large number of iterations until it converges to a stable Nash equilibrium.

In contrast, the *Trajectron++* [98] employs variational learning to improve training convergence and speed. Kernel-based methods employed Mixture Density Networks (MDNs) to build a continuous map capturing the possible motion directions [99] or to learn a multi-model distribution over a set of trajectories [100]. Nevertheless, [101] assumes a time-independent prior over the latent space, and [99, 100] requires a large number of samples to produce distinct trajectories.

3.3 Learning Global Guidance Policies

A key challenge of autonomous navigation in cluttered environments is that the robot's global goal is often located far beyond the planning horizon, meaning that a local subgoal or cost-to-go heuristic must be specified instead. This is straightforward in a static environment (e.g., using euclidean/diffusion [102] distance), but the presence interactive agents makes it difficult to quantify which subgoals will lead to the global goal quickest. We start with reviewing works tackling the navigation problem in crowded environments in Sec. 3.3.1. Then, in Section 3.3.2, we review works employing learning algorithms providing guidance information to MPC-based planning algorithms. In Section 3.3.3 we review approaches combining MPC with RL and its application to autonomous driving in Section 3.3.3.

3.3.1 Navigation Among Crowds

Past work on navigation in cluttered environments often focuses on *interaction models* using geometry [103, 104], physics [105], topologies [106, 107], handcrafted functions [108], and cost functions [109, 109] or joint probability distributions [110] learned from data. While accurate interaction models are critical for collision avoidance, this work emphasizes that the robot's performance (time-to-goal) is highly dependent on the quality of its *cost-to-go model* (i.e., the module that recommends a subgoal for the local planner).

Designing a useful cost-to-go model in this problem remains challenging, as it requires quantifying how "good" a robot's configuration is with respect to dynamic, decision-making agents. In [111], deep RL was introduced as a way of modeling cost-to-go through an offline training phase; the online execution used simple vehicle and interaction models for collision-checking. Subsequent works incorporated other interactions to generate more socially compliant behavior within the same framework [18, 112]. To relax the need for simple online models, [17] moved the collision-checking to the offline training phase. While these approaches use pre-processed information typically available from perception pipelines (e.g., pedestrian detection, tracking systems), other works proposed to learn end-to-end policies [113, 114]. Although all of these RL-based approaches learn to estimate the cost-to-go, the online implementations do not provide guarantees that the recommended actions will satisfy realistic vehicle dynamics or collision avoidance constraints.

3.3.2 Learning-Enhanced MPC

Outside the context of crowd navigation, numerous recent works have proposed learning-based solutions to overcome some of the known limitations of optimization-based methods (e.g., nonlinear MPC) [115]. For example, solvers are often sensitive to the quality of the *initial guess* hence, [116] proposes to learn a policy from data that efficiently "warm-starts" a MPC. *Model inaccuracies* can lead to sub-optimal MPC solution quality; [117] proposes to learn a policy by choosing between two actions with the best expected reward at each timestep: one from model-free RL and one from a model-based trajectory optimizer. Alternatively, RL can be used to optimize the weights of an MPC-based Q-function approximator or to update a robust MPC parametrization [118]. When the model is completely unknown, [119] shows a way of learning a dynamics model to be used in MPC. *Computation time* is another key challenge: [120] learns a cost-to-go estimator to enable shortening of the planning horizons without sacrificing much solution quality, although their approach differs from this work as it uses local and linear function approximators which limits its applicability to high-dimensional state spaces. Learning methods can also be used for *cost tuning*. MPC's cost functions are replaced with a value function learned via RL offline in [121] (terminal cost) and [122] (stage cost).[123] deployed value function learning on a real robot outperforming an expert-tuned MPC.

3.3.3 Combining MPC with RL

Recently, there is increasing interest on approaches combining the strengths of MPC and RL as suggested in [124]. For instance, optimization-based planning has been used to explore high-reward regions and distill the knowledge into a policy neural network, rather than a neural network policy to improve an optimization. [125–127].

Similar to our approach, [128] utilizes the RL policy during training to ensure exploration and employs a MPC to optimize sampled trajectories from the learned policy at test time. Moreover, policy networks have be used to generate proposals for a sampling-based MPC [129], or to select goal positions from a predefined set [130].

Nevertheless, to the extent of our knowledge, approaches combining the benefits of both optimization and learning-based methods were not explored in the context of crowd navigation. Moreover, the works exploring a similar idea of learning a cost-to-go model do not allow to explicitly define collision constraints and ensure safety. Such cost-to-go information can be formulated as learning a value function for the ego-agent state-space providing information which states are more valuable [122].

Application to Autonomous Driving

Recently, there is increasing interest in approaches combining optimization and learning methods [131]. For instance, optimization-based planning has been used to explore high-reward regions and distill the knowledge into a policy neural network [125–127]. For instance, [128] utilizes the RL policy during training to ensure exploration and employs an MPC to optimize sampled trajectories from the learned policy at test time. Similarly, [132] uses RL to learn a driving policy and employs an MPC as a supervisor to ensure safety. Moreover, policy networks have been used to generate proposals for a sampling-based MPC [129] or select goal positions from a predefined set [130].

3.4 Interaction-aware Motion Planning

The literature devoted to the problem of modeling human interactions among traffic participants is vast [8] and includes rule-based, optimization-based, game theoretic and learning-based methods.

3.4.1 Traditional Methods

Traditional autonomous navigation systems typically employ a sequential planning architecture hierarchically decomposing the planning and decision-making pipeline into different blocks such as perception, behavioral planning, motion planning and low-level control [7]. For instance, rule-based methods translate implicit and explicit human-driving behavior into handcrafted functions describing a set of rules directly influencing the motion planning phase. In addition to rules, risk metrics can also be employed to generate cautious driving behavior [133]. For instance, [134] used predictive risk maps to plan the navigation behavior for an AV. These methods have demonstrated excellent ability to solve specific problems (e.g., precedence at an intersection followed by waiting for the availability of enough free space for the vehicle to pass safely) [135–137]. Nevertheless, these methods do not consider the interactions between multiple traffic participants and thus can fail in dense traffic scenarios.

3.4.2 Search-based Methods

The decision-making problem for autonomous navigation is inherently a Partially Observable Markov Decision Process (POMDP) because the other drivers' intentions are not directly observable but can be estimated from sensor data [138]. To improve decision-making and intention estimation, it has been proposed to incorporate the road context information [139]. To deal with a variable number of agents, dimensional reduction techniques have been employed to create a compressed and fixed-size representation of the other agents information [140]. Yet, solving a POMDP online can become infeasible if the right assumptions on the state, action and observation space are not made. For instance, [141] proposed to use Monte Carlo Tree Search (MCTS) algorithms to obtain an approximate optimal solution online and [142] improved the interaction modeling by proposing to feedback the vehicle commands into planning. These methods demonstrated promising results but are limited to environments for which they were specifically designed, demand high computational power and can only consider a discrete set of actions.

3.4.3 Optimization-based Methods

Optimization-based methods are widely used for motion planning since they allow to define collision and kino-dynamics constraints explicitly. These methods include receding-horizon control techniques which allow to plan in real-time and incorporate predicted information by optimizing over a time horizon [8, 143, 144]. However, these works employ simple prediction models and do not consider interaction. Recently, data-driven methods allow to generate interaction-aware predictions [145] that can be used for planning [146, 147]. However, these methods ignore the influence of the ego vehicle's actions in the planning phase struggling to find a collision free trajectory in highly dense traffic scenarios [82]. Not only motion planners must account for the interaction among the driving agents but also generate motions plans which respect social constraints. Hence, to generate socially compatible plans, Inverse Optimal Control techniques have been used to learn human-drivers preferences [148], [149]. These methods either fail to scale to interact with multiple agents [148] or can only handle a discrete set of actions [149] rendering them incapable to be used safely in highly interactive and dense traffic scenarios.

3.4.4 Game Theoretic Methods

Game Theoretic approaches such as [15] model the interaction among agents as a game allowing to infer the influence on each agent's plans. However, the task of modeling interactions requires the inter-dependency of all agents on each other's actions to be embedded within the framework. This results in an exponential growth of interactions as the number of agents increases, rendering the problem computationally intractable. Social value orientation (SVO) is a psychological metric used to classify human driving behavior. [14] models the interaction problem as a dynamic game given the other driver‘s SVO. Similarly, a unscented Kalman filter is used to iteratively update an estimate of the other drivers‘cost parameters [16]. Nevertheless, these approaches require local approximations to find a solution in a tractable manner. Cognitive hierarchy reasoning [150] allows to reduce the complexity of these algorithms by assuming that an agent performs a limited number of iterations of strategic reasoning. For instance, iterative level-k model based on cognitive hierarchy reasoning [150] has been used to obtain a near optimal policy for performing merge maneuvers [151] and lane change [152] in highly dense traffic scenarios. However, these approaches consider a discrete action space and do not scale well with the number of vehicles.

3.4.5 Learning-based Methods

Learning-based approaches leverage on large data collection to build interaction-aware prediction models [145] or to learn a driving policy directly from sensor data [32]. For instance, generative adversarial networks can be used to learn a driving policy imitating human-driving behavior [153]. Conditioning these policies on high-level driving information allows to use it for planning [154]. Moreover, to account for human-robot interaction these policies can be conditioned on the interaction history [155]. Yet, the deployment of these models can lead to catastrophic failures when evaluated in new scenarios or if the training dataset is biased and unbalanced [156].

Reinforcement Learning (RL) has shown great potential for autonomous driving in dense traffic scenarios [157, 158]. For example, DQN has been employed to learn negotiating behavior for lane change [159, 160] and intersection scenarios [161]. Yet, the latter consider a discrete and limited action space. In contrast, in [162] it is proposed to learn a continuous policy (jerk and steering rate) allowing to achieve smooth control of the vehicle. These methods are able to learn a working policy under highly interactive traffic conditions involving multiple entities. However, they fail to provide safety guarantees and reliability, rendering these methods vulnerable to collisions. Recently, a vast amount of works has proposed different ways to introduce safety guarantees of learned RL policies [115]. The key idea behind these works is to synthesize a safety controller when an unsafe action is detected by employing formal verification methods [163], computing offline safe reachability sets [164] or employing safe barrier functions [165]. To reduce conservativeness, [166] proposes to use Linear Temporal Logic to enforce safety probabilistic guarantees. However, *safe RL* methods do not account for interaction among the agents, being highly conservative in dense environments. Finally, [167] learned a decision-making policy to select from a discrete and limited set of predefined constraints which ones to enable in an MPC formulation and thus, controlling the vehicle behavior applied to intersection scenarios.

3.5 Conclusions

The previous sections have introduced many works proposing new motion planning, prediction, and decision-making techniques. Here, we provide some overall conclusions on the most promising methods used to solve the motion planning, prediction, and decision-making problems.

Related to motion planning, modern solvers enabled optimization methods to perform real-time motion planning [168]. Therefore, techniques such as Model Predictive Control [21, 169] became highly promising, allowing to incorporate constraints, generate kino-dynamically feasible solutions and incorporate prediction information in the planning stage.

On the trajectory prediction problem, over the past decade, Deep Learning (DL) models [34] (i.e., neural networks) have dominated the field enabling us to learn prediction models incorporating high-dimensional information from large amounts of data [170]. Moreover, in combination with variational Bayesian learning methods [171], neural networks are a powerful tool to learn complex probability models from data.

Finally, to tackle the decision-making problem, Reinforcement Learning [38] methods are widely employed due to their ability to lean policy optimizing long-term rewards. Furthermore, in combination with high capacity models such as neural works (i.e., Deep Reinforcement Learning (DRL) [172]) methods have demonstrated to achieve *super human* capabilities in many complex problems [173].

Hence, the methods developed in this thesis build upon three main pillars: MPC, DL, and DRL.

4

Model Predictive Contouring Control for Collision Avoidance in Dynamic Environments

---

This chapter is based on:

- B. Brito, B. Floor, L. Ferranti and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, 4(4): 4459-4466, 2019
- L. Ferranti, B. Brito, E. Pool, Y. Zheng, R. Ensing, R. Happee, B. Shyrokau, J. Kooij, J. Alonso-Mora and D. Gavrila, "SafeVRU: A research platform for the interaction of self-driving vehicles with vulnerable road users," *IEEE Intelligent Vehicles Symposium (IV)* (pp. 1660-1666), 2019.

Code: https://github.com/tud-amr/amr-lmpcc

4.1 Introduction

Applications where autonomous ground vehicles (AGVs) closely navigate with humans in complex environments require the AGV to safely avoid static and moving obstacles while making progress towards its goal. Motion planning and control for AGVs are typically addressed as two independent problems [8]. In particular, the motion planner generates a collision-free path and the motion controller tracks such a path by directly commanding the AGV's actuators. Our method combines both motion planning and control in one module, relying on constrained optimization techniques, to generate kinematically feasible local trajectories with fast replanning cycles. More specifically, we rely on a Model Predictive Controller (MPC) to compute an optimal control command for the controlled system, which directly incorporates the predicted intentions of dynamic obstacles. Consequently, it reacts in advance to smoothly avoid moving obstacles.

Here, we propose a practical reformulation of the Model Predictive Contouring Control (MPCC) approach [58], namely, a Local Model Predictive Contouring Control (LMPCC) approach, suitable for real-time collision-free navigation of AGVs in complex environments with several agents. We build on [58], with the following contributions to make the design applicable to mobile robots navigating in unstructured environments with humans:

- A static obstacle avoidance strategy that explicitly constrains the robot's positions along the prediction horizon to a polyhedral approximation of the collision-free area around the robot.
- A closed-form bound to conservatively approximate collision avoidance constraints that arise from ellipsoidal moving obstacles.
- A fully integrated MPCC approach that runs in real-time on-board of the robot and with on-board perception.

We present experimental results with a mobile robot navigating in indoor environments among static and moving obstacles and compare them with three state-of-art planners, namely the dynamic window [11], a classical MPC for tracking [52] and a socially-aware motion planner [174]. Our design is fully implemented on board of the robot including localization and environment perception, i.e., detection of static obstacles and pedestrians. Our design runs in real time thanks to its lightweight implementation. Moreover, in Section 4.5 we present experimental results with a real autonomous vehicle. The proposed method has been open-sourced[1].

---

[1] https://github.com/tud-amr/amr-lmpcc

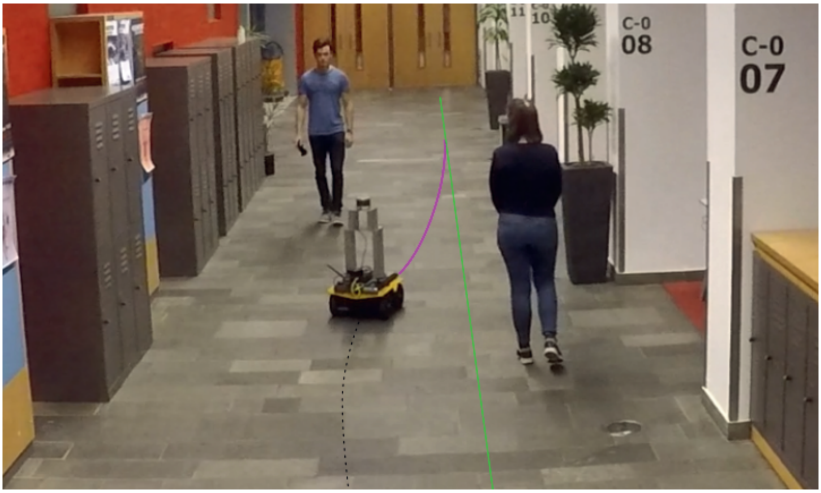


Figure 4.1: The faculty corridor was the scenario used to evaluate the capabilities of our method to avoid dynamic obstacles.

4.2 Preliminaries

4.2.1 Robot Description

Let B denote an AGV on the plane $\mathcal{W} = \mathbb{R}^2$. The AGV dynamics are described by the discrete-time nonlinear system

$$\mathbf{z}(t+1) = f(\mathbf{z}(t), \mathbf{u}(t)), \tag{4.1}$$

where $\mathbf{z}(t)$ and $\mathbf{u}(t)$ are the state and the input of the robot, respectively, at time $t \geq 0$[2]. For the case of our mobile robot we consider the state to be equal to the configuration, that is, $\mathbf{z}(t) \in \mathcal{C} = \mathbb{R}^2 \times \mathcal{S}$. For the case of a car (Sec. 4.5), the state space includes the speed of the car. The area occupied by the robot at state $\mathbf{z}$ is denoted by $\mathcal{B}(\mathbf{z})$. which is approximated by a union of n_c circles, i.e., $\mathcal{B}(\mathbf{z}) \subseteq \bigcup_{c \in \{1,\dots,n_c\}} \mathcal{B}_c(\mathbf{z}) \subset \mathcal{W}$. The center of each circle, in the inertial frame, is given by $\mathbf{p} + R_B^W(\mathbf{z})\mathbf{p}_c^B$. Where $\mathbf{p}$ is the position of the robot (extracted from $\mathbf{z}$), $R_B^W(\mathbf{z})$ is the rotation matrix given by the orientation of the robot, and $\mathbf{p}_c^B$ is the center of circle c expressed in the body frame.

4.2.2 Static Obstacles

The static obstacle environment is captured in an occupancy grid map, where the area occupied by the static obstacles is denoted by $\mathcal{O}^{\text{static}} \subset \mathcal{W}$. In our experiments we consider both a global map, which is built a priori and used primarily for localization, and a local map from the current sensor readings. Therefore, the static map is continuously updated

---

[2] In the remainder of the chapter we omit the time dependency when it is clear from the context.

locally. Dynamic obstacles, such as people, that are recognized and tracked by the robot are removed from the static map and considered as moving obstacles.

4.2.3 Dynamic Obstacles

Each moving obstacle i is represented by an ellipse of area $\mathcal{A}_i \subset \mathcal{W}$, defined by its semi-major axis a_i, its semi-minor axes b_i, and a rotation matrix $R_i(\psi)$. We consider a set of moving obstacles $i \in \mathcal{I} := \{1, \dots, n\}$, where n can vary over time. The area occupied by all moving obstacles at time instant t is given by $\mathcal{O}_t^{\text{dyn}} = \bigcup_{i\in\{1,\dots,n\}} \mathcal{A}_i(\mathbf{z}_i(t))$, where $\mathbf{z}_i(t)$ denotes the state of moving obstacle i at time t. In this work, and without a loss of generality, we assume a constant velocity model with Gaussian noise $\omega_o(t) \sim \mathcal{N}(0, Q_o(t))$ in acceleration, that is, $\ddot{\mathbf{p}}_i(t) = \omega_i(t)$, where $\mathbf{p}_i(t)$ is the position of obstacle i at time t. Given the measured position data of each obstacle, we estimate their future positions and uncertainties with a linear Kalman filter [175].

4.2.4 Global Reference Path

Consider that a reference path is available. In its simplest form, the reference path can be a straight line to the goal position or a straight line in the direction of preferred motion. But it could also be given by a global planner. We consider a global reference path $\mathcal{P}$ consisting of a sequence of path segments connecting M way-points $\mathbf{p}_m^r = [x_m^p, y_m^p] \in \mathcal{W}$ with $m \in \mathcal{M} := \{1, \dots, M\}$. For smoothness, we consider that each path segment $\varsigma_m(\theta)$ is defined by a cubic polynomial. We denote by θ a variable that (approximately) represents the traveled distance along the reference path, and which is described in more detail in Sec. 4.3.3. We do not require the reference to be collision free, therefore, the robot may have to deviate from it to avoid collisions.

4.2.5 Problem Formulation

The objective is to generate, for the robot, a collision free motion for N time-steps in the future, while minimizing a cost function J that includes a penalty for deviations from the reference path. This is formulated in the optimization problem

$$
\begin{aligned}
J^* = & \min_{\mathbf{z}_{0:N}, \mathbf{u}_{0:N-1}, \theta_{0:N-1}} \sum_{k=0}^{N-1} J(\mathbf{z}_k, \mathbf{u}_k, \theta_k) + J(\mathbf{z}_N, \theta_N) \\
\text{s.t.} \quad & \mathbf{z}_{k+1} = f(\mathbf{z}_k, \mathbf{u}_k), \ \theta_{k+1} = \theta_k + v_k \tau, && (4.2a) \\
& \mathcal{B}(\mathbf{z}_k) \cap \left(\mathcal{O}^{\text{static}} \cup \mathcal{O}_k^{\text{dyn}}\right) = \emptyset, && (4.2b) \\
& \mathbf{u}_k \in \mathcal{U}, \ \mathbf{z}_k \in \mathcal{Z}, \ \mathbf{z}_0, \theta_0 \text{ given.} && (4.2c)
\end{aligned}
$$

Where v_k is the forward velocity of the robot (for the mobile robot it is part of the input and for the car it is part of the state), τ is the time-step and $\mathcal{Z}$ and $\mathcal{U}$ are the set of admissible states and inputs, respectively. $\mathbf{z}_{1:N}$ and $\mathbf{u}_{0:N-1}$ are the set of states and control inputs, respectively, over the prediction horizon T_{horizon}, which is divided into N prediction steps. θ_k denotes the predicted progress along the reference path at time-step k. By solving the optimization problem, we obtain a locally optimal sequence of commands $[\mathbf{u}_t^*]_{t=0}^{t=N-1}$ to

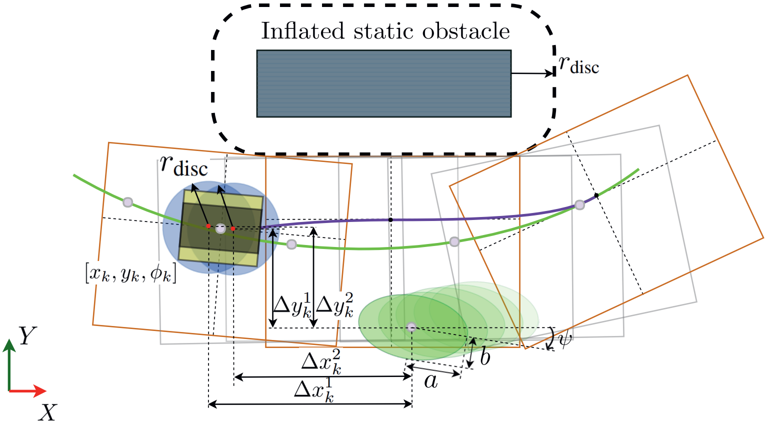


Figure 4.2: Representation of the convex free space (orange squares) around each prediction step on the prediction horizon (purple line) with respect to the inflated static environment and the collision space of the dynamic obstacle (green ellipses) with respect to the vehicle representing discs (blue).

guide the robot along the reference path while avoiding collisions with static and moving obstacles.

4.3 Method

The proposed method consists of the following steps, which are executed in every planning loop.

1. Search for a collision-free region in the updated static map centered on the robot and constrain the control problem such that the robot remains inside;
2. Predict the future positions of the dynamic obstacles and use the corrected bound to ensure dynamic collision avoidance;
3. Solve a modified MPCC formulation applicable to mobile robots (Section 4.3.3).

4.3.1 Static Collision Avoidance

Given the static map of the environment, we compute a set of convex four-sided polygons in free space. This representation can provide larger collision-free areas compared with other approximations such as circles. To obtain the set of convex regions at time t, we first shift the optimal trajectory computed at time $t-1$, namely, $\boldsymbol{q}_{0:N} = [\boldsymbol{p}^*_{1:N|t-1}, \boldsymbol{q}_N]$, where $\boldsymbol{q}_N$ is a extrapolation of the last two points, that is, $\boldsymbol{q}_N = 2\boldsymbol{p}^*_{N|t-1} - \boldsymbol{p}^*_{N-1|t-1}$. Then, for each point $\boldsymbol{q}_k$ ($k = 1, \dots, N$) we compute a convex region in free space, given by a set of four linear constraints $c_k^{\text{stat}}(\boldsymbol{p}_k) = \bigcup_{l=1}^{4} c_k^{\text{stat},l}(\boldsymbol{p}_k)$. This region separates $\boldsymbol{q}_k$ from the closest obstacles.

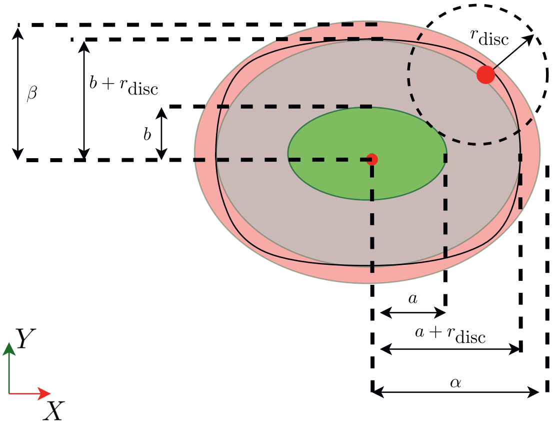


Figure 4.3: Safety boundary computation. The ellipsoid with semi-major axis and semi-minor axis, a and b respectively, is represented in green, the ellipsoid with axis enlarged by r_{disc} is represented in gray, the Minkowsky sum is represented in black and the ellipsoid enlarged by δ is represented in red.

In our implementation we compute a rectangular region aligned with the orientation of the trajectory at $\boldsymbol{q}_k$, where each linear constraint is obtained by a search routine and reduced by the radius of the robot circles r_{disc}. Figure 4.2 shows the collision-free regions along the prediction horizon defined as yellow boxes. At prediction step k for a robot with state $\boldsymbol{z}$ the resulting constraint for disc j and polygon side l is

$$c^{\text{stat},l,j}(\boldsymbol{z}) = h^l - \vec{n}^l \cdot \left(\boldsymbol{p} - R_B^W(\boldsymbol{z})\boldsymbol{p}_j^B\right) > 0, \tag{4.3}$$

where h^l and $\vec{n}^l$ define each side of the polygons.

4.3.2 Dynamic Collision Avoidance

Recall that each moving obstacle i is represented by its position $\boldsymbol{p}_i(t)$ and an ellipse of semi-axis a_i and b_i and a rotation matrix $R_i(\psi)$. For each obstacle $i \in \{1, \ldots, n\}$, and prediction step k, we impose that each circle j of the robot does not intersect with the elliptical area occupied by the obstacle. Omitting i for simplicity, the inequality constraint on each disc of the robot with respect to the obstacles is

$$c_k^{\text{obst},j}(\boldsymbol{z}_k) = \begin{bmatrix} \Delta x_k^j \\ \Delta y_k^j \end{bmatrix}^{\mathrm{T}} R(\psi)^{\mathrm{T}} \begin{bmatrix} \frac{1}{\alpha^2} & 0 \\ 0 & \frac{1}{\beta^2} \end{bmatrix} R(\psi) \begin{bmatrix} \Delta x_k^j \\ \Delta y_k^j \end{bmatrix} > 1, \tag{4.4}$$

where the distance between disc j and the obstacle is separated into its Δx^j and Δy^j components (Figure 4.2). The parameters α and β are the semi-axes of an enlarged ellipse that includes the union of the original ellipse and the circle.

While previous approaches approximated the Minkowski sum of the ellipse with the circle as an ellipse of semi-major $\alpha = a + r_{\text{disc}}$ and semi-minor axis $\beta = b + r_{\text{disc}}$ [58], this assumption is not correct and collisions can still occur [176]. We now describe how to compute the values for α and β such that collision avoidance is guaranteed, represented by the larger in light red ellipsoid in Fig. 4.3.

Consider two ellipsoids $E_1 = \mathrm{Diag}(\frac{1}{a^2}\frac{1}{b^2})$ and $E_2 = \mathrm{Diag}\left(\frac{1}{(a+\delta)^2}, \frac{1}{(b+\delta)^2}\right)$. E_1 is an ellipsoid with a and b as semi-major and semi-minor axes, respectively. E_2 represents the ellipsoid E_1 enlarged by δ in both axis. The goal is to find the smallest ellipsoid that bounds the Minkowsky sum. This is equivalent to find the minimum value of δ such that the minimum distance between ellipsoid E_1 and E_2 is bigger than r[3], the radius of the circle bounding the robot.

Lemma 1 ([177]). *Let $X^TA_1X = 1$ and $X^TA_2X = 1$ be two quadratics in R^n. Iff the matrix $A_1 - A_2$ is sign definite, then the square of the distance between the quadratic $X^TA_1X = 1$ and the quadratic $X^TA_2X = 1$ equals the minimal positive zero of the polynomial.*

$$F(z) = D_\lambda(\det \lambda A_1 + (z - \lambda A_2) - \lambda(z - \lambda A_1 A_2))$$

where D stands for the discriminant of the polynomial treated with respect to λ.

Considering $A_1 = E_1$, $A_2 = E_2$ and $\{\delta, a, b, \} \in \mathrm{R}^+$ this ensures that $E_1 - E_2$ is sign definite. Hence, we can apply Lemma 1 to determine the polynomial $F(z)$ and its roots. For the two ellipsoids E_1 and E_2, the roots λ of $F(z)$ are:

$$\lambda \in \left\{ \begin{array}{r} \dfrac{(2a(r+\delta)^3 + 2b(r+\delta)^3 + 4ab(r+\delta)^2}{(a^2 + 2ab + 2ra + b^2 + 2rb}, \\ (r+\delta)^2, \\ 4a^2 + 4a(r+\delta) + (r+\delta)^2, \\ 4b^2 + 4b(r+\delta) + (r+\delta)^2 \end{array} \right\}. \tag{4.5}$$

The first two roots have multiplicity two. The minimum distance equation is the square root of the minimal positive zero of $F(z)$. Thus, the minimum enlargement factor is found by solving for the value of δ that satisfies $\min_j \lambda(j) = r^2$.

A closed form formula can be obtained by noting that the $\min_j \lambda(j)$ is achieved for the first root and solving this equation. Due to its length, it is not presented in this paper but can be found at [4]. This value of semi-axis $\alpha = a + \delta$ and $\beta = b + \delta$ guarantee that the constraint ellipsoid entirely bounds the collision space.

---

[3] Note we use r instead of r_{disc} in the reminder of the section to simplify the notation.

[4] A Mathemathica notebook with the derivation of the bound and a Matlab script as example of its computation can be found in http://www.alonsomora.com/docs/19-debrito-boundcomputation.zip.

4.3.3 Model Predictive Contouring Control

MPCC is a formulation specially tailored to path-following problems. This section presents how to modify the baseline method [58] and make it applicable to mobile robots navigating in unstructured environments with on-board perception.

Progress on Reference Path

Eq. 4.2 approximates the evolution of the path parameter by the traveled distance of the robot. In each planning stage we initialize θ_0. We find the closest path segment, denoted by m, and compute the value of θ_0 via a line search in the neighborhood of the previously predicted path parameter.

Selecting the Number of Path Segments

As detailed in Section 4.2.4, the *global* reference is composed of M path segments. To lower the computational load, only $\eta \leq M$ path segments are used to generate the *local* reference that is incorporated into the optimization problem. The number of path segments η cannot be arbitrarily small, and there is a minimum number of segments to be selected to ensure the robot follows the reference path along a prediction horizon. The number of path segments η in the local reference path is a function of the prediction horizon length, the individual path segment lengths, and the speed of the robot at each time instance. We select a conservative η by considering the maximum longitudinal velocity $v_{\max}$ and imposing that the covered distance is lower than a lower bound of the travelled distance along the reference path, namely,

$$\underbrace{\tau \sum_{j=0}^{N-1} v_j}_{\text{Traveled dist.}} \leq \tau N v_{\max} \leq \underbrace{\sum_{i=m+1}^{m+\eta} \|\boldsymbol{p}^{\mathrm{r}}_{i+1} - \boldsymbol{p}^{\mathrm{r}}_{i}\|}_{\text{Waypoints dist.}} \leq \underbrace{\sum_{i=m+1}^{m+\eta} s_i}_{\text{Ref. path length}} , \tag{4.6}$$

where m is the index of the closest path segment to the robot, τ is the length of the discretization steps along the horizon, and s_i the length of each path segment.

Maintaining Continuity over the Local Reference Path

We concatenate the η reference path segments into a differentiable local reference path $\boldsymbol{L}^r$, which will be tracked by the LMPCC, as follows

$$\bar{\boldsymbol{p}}^r(\theta_k) = \sum_{i=m}^{m+\eta} \sigma_{i,+}(\theta_k)\sigma_{i,-}(\theta_k)\varsigma_i(\theta_k), \tag{4.7}$$

where $\sigma_{i,-}(\theta_k) = 1/(1 + e^{(\theta - \sum_{j=m}^{i} s_i)/\epsilon})$ and $\sigma_{i,+}(\theta_k) = 1/(1 + e^{(-\theta + \sum_{j=m}^{i-1} s_i)/\epsilon})$ are two sigmoid activation functions for each path segment and ϵ is a small design constant. This representation ensures a continuous representation of the local reference path needed to compute the solver gradients.

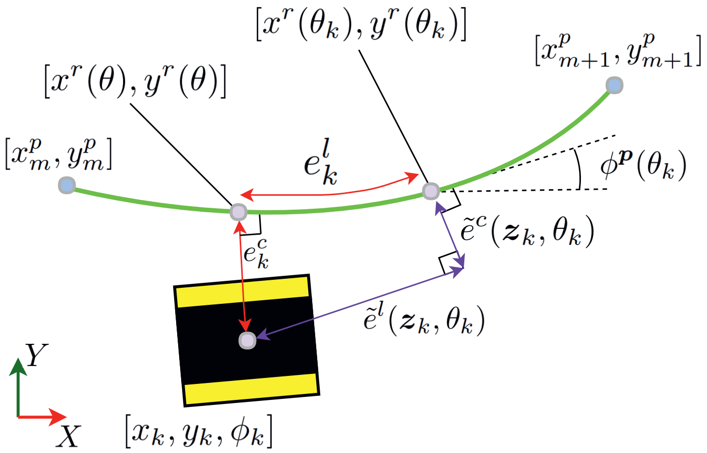

$[x^r(\theta_k), y^r(\theta_k)]$
$[x^r(\theta), y^r(\theta)]$
$[x^p_{m+1}, y^p_{m+1}]$
$[x^p_m, y^p_m]$
e^l_k
$\phi^p(\theta_k)$
$\tilde{e}^c(\boldsymbol{z}_k, \theta_k)$
e^c_k
$\tilde{e}^l(\boldsymbol{z}_k, \theta_k)$
Y
X
$[x_k, y_k, \phi_k]$

Figure 4.4: Approximated contour and lag errors on the path segment.

Cost Function

For tracking of the reference path, a contour and a lag error are defined, see Figure 4.4 and combined in an error vector $\boldsymbol{e}_k := [\tilde{\epsilon}^c(\boldsymbol{z}_k, \theta_k), \tilde{\epsilon}^l(\boldsymbol{z}_k, \theta_k)]^{\mathrm{T}}$, with

$$\boldsymbol{e}_k = \begin{bmatrix} \sin\phi(\theta_k) & -\cos\phi(\theta_k) \\ -\cos\phi(\theta_k) & -\sin\phi(\theta_k) \end{bmatrix} \left(\boldsymbol{p}_k - \bar{\boldsymbol{p}}^r(\theta_k)\right), \tag{4.8}$$

where $\phi(\theta_k) = \arctan(\partial y^r(\theta_k)/\partial x^r(\theta_k)$ is the direction of the path. Consequently, the LMPCC tracking cost is

$$J_{\text{tracking}}(\boldsymbol{z}_k, \theta_k) = \boldsymbol{e}_k^T Q_\epsilon \boldsymbol{e}_k, \tag{4.9}$$

where Q_ϵ is a design weight.

The solution which minimizes the quadratic tracking cost defined in Eq. 4.9 drives the robot towards the reference path. To make progress along the path we introduce a cost term that penalizes the deviation of the robot velocity v_k from a reference velocity v_{ref}, i.e., $J_{\text{speed}}(\boldsymbol{z}_k, \boldsymbol{u}_k) = Q_v(v_{\text{ref}} - v_k)^2$ with Q_v a design weight. This reference velocity is a design parameter given by a higher-level planner and can vary across path reference segments.

To increase the clearance between the robot and moving obstacles, we introduce an additional cost term similar to a potential function,

$$J_{\text{repulsive}}(\boldsymbol{z}_k) = Q_R \sum_{i=1}^{n} \left(\frac{1}{(\Delta x_k)^2 + (\Delta y_k)^2 + \gamma} \right) \quad , \tag{4.10}$$

where Q_R is a design weight, and Δx_k, Δy_k represent the components of the distance from the robot to the dynamic obstacles. A small value $\gamma \geq 0$ is introduced for numerical stability.

Algorithm 4 Local Model Predictive Contouring Control

```
1:  Given $z_{\text{init}}$, $z_{\text{goal}}$, $\mathcal{O}^{\text{static}}$, $\mathcal{O}_0^{\text{dyn}}$, and $N$
2:  for $t = 0, 1, 2, ...$ do
3:      $z_0 = z_{\text{init}}$
4:      Estimate $\theta_0$ according to Section 4.3.3
5:      Select $\eta$ according to Equation (4.6)
6:      Build $\bar{\boldsymbol{p}}^r(\theta_k)$, $k = 1, ..., N$, according to Equation (4.7)
7:      Compute $c_k^{\text{stat,j}}(\boldsymbol{p}_k)$ along $\boldsymbol{q}_{0:N}$
8:      Get dynamic-obstacles predicted pose (Sec. 4.3.2)
9:      Solve the optimization problem of Equation (4.11)
10:     Apply $\boldsymbol{u}_0^*$
11: end for
```

Eq. 4.10 adds clearance with respect to obstacles and renders the method more robust to localization uncertainties. Additionally, we penalize the inputs with $J_{\text{input}}(z_k, \theta_k) = \boldsymbol{u}_k^T Q_u \boldsymbol{u}_k$, where Q_u is a design weight.

The LMPCC control problem is then given by a receding horizon nonconvex optimization, formally,

$$
\begin{aligned}
J^* = \min_{z_{0:N}, \boldsymbol{u}_{0:N-1}, \theta_{0:N}} & \sum_{k=0}^{N-1} J(\mathbf{z}_k, \mathbf{u}_k, \theta_k) + J(z_N, \theta_N) && \text{(4.11a)} \\
\text{s.t. :} \ & (4.2\text{a}), (4.2\text{c}), && \text{(4.11b)} \\
& c_k^{\text{stat},l,j}(z_k) > 0, \forall j \in \{1, \ldots, n_c\}, l \in \{1, ..., 4\} && \text{(4.11c)} \\
& c_k^{\text{obst},j}(z_k) > 1, \forall j \in \{1, \ldots, n_c\}, \forall \, \text{obst} && \text{(4.11d)}
\end{aligned}
$$

where the stage cost is $J(\mathbf{z}_k, \mathbf{u}_k, \theta_k) := J_{\text{tracking}}(z_k, \theta_k) + J_{\text{speed}}(z_k, \boldsymbol{u}_k) + J_{\text{repulsive}}(z_k) + J_{\text{input}}(\boldsymbol{u}_k)$ and the terminal cost is $J(z_N, \theta_N) := J_{\text{tracking}}(z_N, \theta_N) + J_{\text{repulsive}}(z_N)$. Eq. 11c and Eq. 11d are defined by Eq. 4.3 and Eq. 4.4, respectively. Algorithm 4 summarizes our method. Note that at each control iteration, we solve (using [3]) Problem (4.11) until either a Karush-Kuhn-Tucker condition [27] or the maximum number of iterations ($\texttt{iter}_{\text{max}}$) is satisfied (line 9). The selection of $\texttt{iter}_{\text{max}}$ is empirically based on the maximum number of iterations allowed within the sampling time of our system to guarantee real-time performance.

4.4 Results - Mobile Robot

This section presents experimental and simulation results for three scenarios with a mobile robot. We evaluate different settings for the parameters in our planner (Section 4.4.2), as well as compare its performance in static (Section 4.4.3) and dynamic (Section 4.4.4) environments against state of art motion planners [11, 52, 174]. In Section 4.5, we present experimental results on a real autonomous vehicle.

4.4.1 Experimental Setup

Hardware Setup

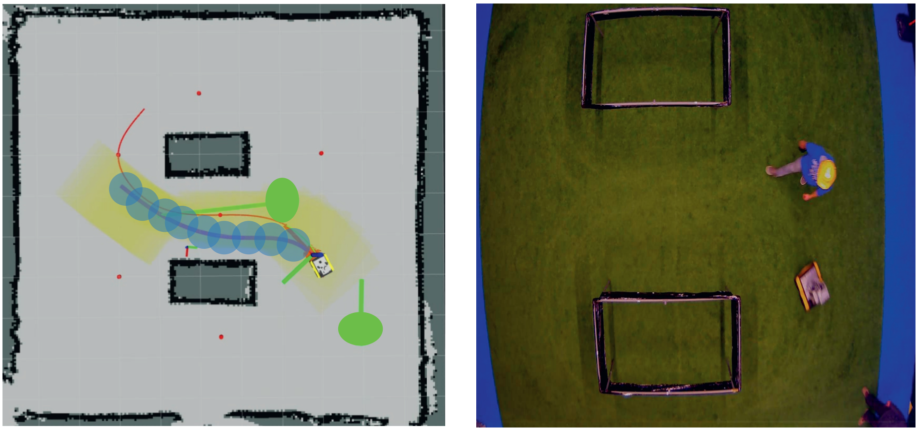
Our experimental platform is a fully autonomous Clearpath Jackal ground robot, for which we implemented on-board all the modules used for localization, perception, motion planning, and control. Our platform is equipped with an Intel i5 CPU@2.6GHz, which is used to run the localization and motion planning modules, a Lidar Velodyne for perception, and an Intel i7 NUC mini PC to run the pedestrian tracker.

Software Setup

To build the global reference path we first define a series of waypoints and we construct a smooth global path by connecting the waypoints with a clothoid. We then sample intermediate waypoints from this global path and connect them with 3rd order polynomials, which are then used to generate the local reference path.

The robot localizes with respect to a map of the environment, which is created before the experiments. For static collision avoidance the robot utilizes a map that is updated online with data from its sensors and we employ a set of rectangles to model the free space. Our search routine expands the sides of a vehicle-aligned rectangle simultaneously in the occupancy grid environment with steps of $\Delta^{\text{search}} = 0.05\text{m}$, until either an occupied cell is found or the maximum search distance $\Delta^{\text{search}}_{\text{max}} = 2\text{m}$ is reached. Once an expanding rectangle side is fixed as a result of an occupied cell, the rest of the rectangle sides are still expanded to search for the largest possible area. This computation runs in parallel to the LMPCC solver, in a different thread, and with the latest available information. Our experiments employ the open-source SPENCER Pedestrian tracker and 2d laser data [178] for detection and tracking of dynamic obstacles. If a pedestrian is detected, it is removed from the static map and treated as a moving ellipse. Our simulations use the open-source ROS implementation of the Social Forces model [105] for pedestrian simulation.

The LMPCC problem of Eq. 4.11 is nonconvex. Our planner solves this problem online in real time using ACADO [179] and its C-code generation tool. We use a continuous-time kinematic unicycle motion model [180] to describe the robot's kinematics. The model is then discretized directly in ACADO using a multiple-shooting method combined with a Gauss-Legendre integrator of order 4, no Hessian approximations, and a sampling time of 50 ms. We select qpOASES [181] to solve the resulting QP problem and set a KKT tolerance of 10^{-4} and a maximum of 10 iterations. If no feasible solution is found within the maximum number of iterations, then the robot decelerates. The planner computes a new solution in the next cycle. Based on our experience, this allowed to recover the feasibility of the planner quickly. Our motion planner is implemented in C++/ROS and will be released open source.



a) Visualization *b) Top view*

Figure 4.5: Experimental setup considering perfect perception and localization using a motion capture system for performance evaluation.

4.4.2 Parameter Evaluation

To evaluate the performance of the planner we performed several experiments at different reference speeds, $v_{\text{ref}} \in \{1\,\text{m/s}, 1.25\,\text{m/s}, 1.5\,\text{m/s}\}$, and prediction-horizon lengths, $T_{\text{Horizon}} \in \{1\,\text{s}, 3\,\text{s}, 5\,\text{s}\}$. The robot follows a figure-8 path (red line in Figure 4.5-(a)) while avoiding two pedestrians (green ellipses) and staying within the collision-free area (yellow rectangles). We use one circle to represent the planned position of the robot (light blue circles). Each pedestrian is bounded by an ellipse of semi-axis 0.3 m and 0.2 m. The predicted positions of the pedestrians are represented by a green line. We align the semi-minor axis to the pedestrian walking direction. For this experiment we rely on a motion capture system to obtain the position of the obstacles and robot. Figure 4.6 shows the computation time to solve the optimization problem. For $v_{\text{ref}} \in \{1, 1.25, 1.5\}$ m/s and $T_{\text{Horizon}} \in \{1, 3\}$ s the computation times are under 50 ms, which is lower than the cycle-time defined for the planner. But, for $T_{\text{Horizon}} = 5$ s the 99th percentile is above the 50ms, not respecting the real-time constraint. The cases in which the planner exceeds the sampling time of the system are the situations in which the pedestrians suddenly step in front of the robot or change their direction of motion, requiring the solver more iterations to find a feasible solution. If not all the constraints can be satisfied, our problem becomes infeasible, and no solution is found. In this case, we reduce the robot velocity, allowing the solver to recover the feasibility after few iterations. Table 4.1 summarizes the behavior of the planner in terms of clearance, that is, the distance from the border of the circle of the robot to the border of the ellipse defining the obstacles. It demonstrates that a short horizon leads to

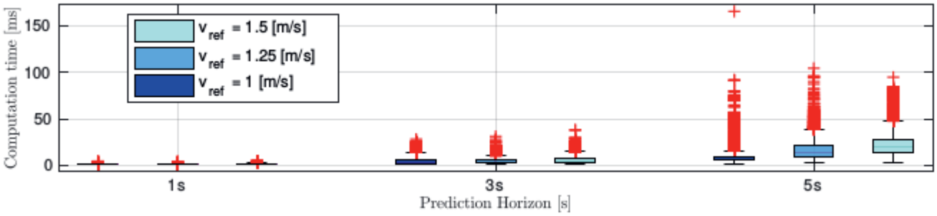


Figure 4.6: Computation time required to solve the LMPCC problem online for different velocity references and horizon lengths. The central mark indicates the median. The bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The red crosses represent the outliers.

Table 4.1: Clearance between the robot and the dynamic obstacles for different velocity references and horizon lengths.

Prediction Horizon [s]	Clearance Mean (1st percentile) [m]		
	v_{ref} = 1	v_{ref} = 1.25	v_{ref} = 1.5
1	1.54 (0.077)	1.60 (0.025)	1.69 (0.003)
3	1.66 (0.077)	1.69 (0.072)	1.82 (0.065)
5	1.69 (0.062)	1.68 (0.055)	1.92 (0.043)

lower safety distances and more importantly, that our method was able to keep a safe clearance in most cases.

Based on these results, we selected a reference speed of 1.25 m/s and a horizon length of 3 s for the following experiments.

4.4.3 Static Collision Avoidance

In this experiment we compare the proposed planner with two baseline approaches:

- *A MPC tracking controller* [52]. We minimize the deviation from positions on the reference path up to 1 m ahead of the robot.
- *The Dynamic Window (DW)* [11]. We use the open-source ROS stack implementation. The DW method receives the next waypoint once the distance to the current waypoint is less than 1 m.

For this and the following experiment, we fully rely on the onboard localization and perception modules. A VLP16 Velodyne Lidar is used to build and update a local map centered in the robot for static collision avoidance. The prebuilt offline map is only used for localization.

In this experiment the mobile robot navigates along a corridor while tracking a global reference path (red waypoints in Figure 4.7). When it encounters automatic doors, the

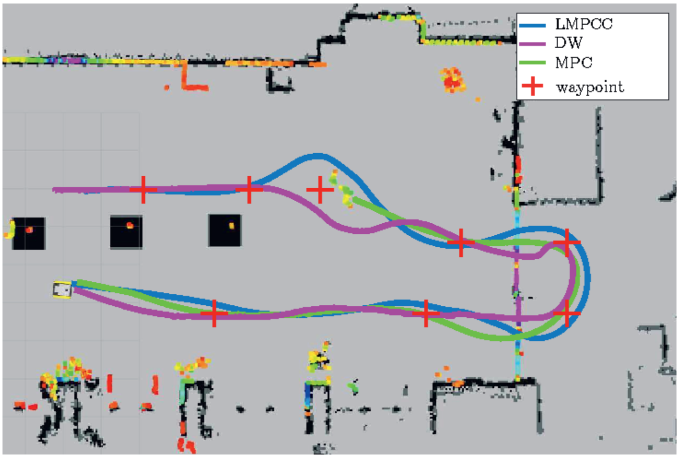


Figure 4.7: Static collision avoidance scenario. The red crosses depict the path to follow (waypoints) and the colored points are the laser scan data. The blue, green, and magenta lines represent the trajectory obtained with the LMPCC,the MPC tracking controller, and the Dynamic Window, respectively.

robot must wait for them to open. When it encounters the obstacle located near the third upper waypoint from the left, which was not in the map, the robot must navigate around it. Figure 4.7 shows the results of this experiment. All the three approaches are able to follow the waypoints and interact with the automatic door. When they encounter the second obstacle the classical MPC design fails to proceed towards the next waypoint. The DW and the LMPCC approaches are able to complete the task. Both methods showed similar performance (e.g., the traveled distance was 30.59 m for our LMPCC and 30.61 m for the DW). The time to the goal was relatively higher for the DW (50 s) compared to the LMPCC (41 s). This difference is mainly due to the ability of the LMPCC to follow a reference velocity. In addition, we tested DW and LMPCC in a scenario where half of the first door does not open due to malfunction. In such scenario, the LMPCC was able to traverse around the broken door while the DW gets into a deadlock state due to the narrow opening.

4.4.4 Dynamic Collision Avoidance

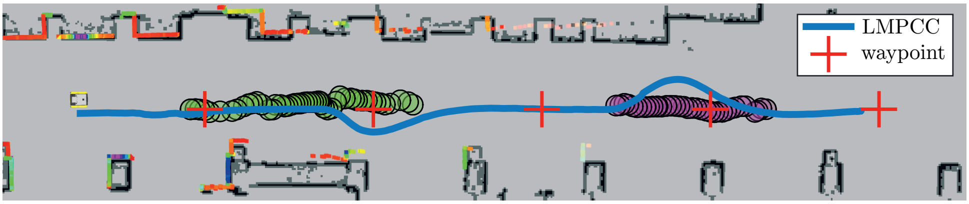
Our simulations compare our planner with an additional baseline: *a socially-aware motion planner* named Collision Avoidance with Deep RL (CADRL) [174]. We employ the open-source ROS stack implementation. The CADRL method receives the same waypoints as the DW method.

Simulation Results

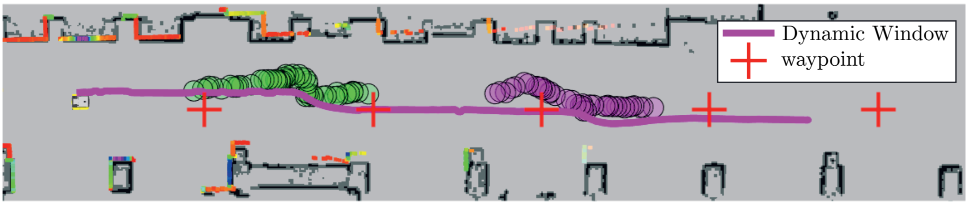
We compare the performance of the MPCC planner with the DW and CADRL baselines in the presence of pedestrians. Figure 4.1 shows the setup of our experiment. To avoid the overlap of the static and dynamic collision constraints, the detected pedestrians are removed from the updated map used for static avoidance and modeled as ellipses with a constant velocity estimate. The global path, consists of a straight line along the corridor. The robot has to follow this path while avoiding collisions with several pedestrians moving in the same or opposite direction. In this experiment, we do not evaluate the MPC tracking controller since it was unable to complete the previous experiment. Aggregated results in Table 4.2 show that the LMPCC outperforms the other methods. It achieves a considerably lower failure rate, smaller traveling distances, and maintains larger safety distances to the pedestrians. Only for the four pedestrians case, the DW achieved larger mean clearance, but with larger standard deviation. By accounting for the predictions of the pedestrians, our method can react faster and thus generate safer motion plans. Table 4.2 also shows that the number of failures grows with the number of agents. Yet, our method can scale up to six pedestrians with low collision probability and perform real-time. For larger crowds our method would select the closer 6 pedestrians.

Table 4.2: Statistic results of minimum distance to the pedestrians (where clearance is defined as the border to border distance), traveling distance and percentage of failures obtained for 100 random test cases of the dynamic collision avoidance experiment for $n \in \{2, 4, 6\}$ *agents. The pedestrians follow the social forces model [105].*

# agents	Clearance Mean (1st percentile) [m]			% failures (% collisions / % stuck)			Traveled distance Mean (Std.) [m]		
	DW	CADRL	LMPCC	DW	CADRL	LMPCC	DW	CADRL	LMPCC
2	0.28 (0)	0.15 (0)	**0.29** (0.015)	20 (19 / 1)	4 (4 / 0)	**2** (0 / 2)	17.99 (3.9)	19.27 (3.7)	**15.81** (3.2)
4	**0.46** (0)	0.32 (0)	0.25 (0.026)	35 (32 / 3)	31 (31 / 0)	**5** (2 / 3)	19.43 (5.6)	21.34 (4.4)	**15.77** (4.4)
6	0.06 (0)	0.33 (0)	**0.38** (0.013)	43 (41 / 2)	51 (51 / 0)	**7** (5 / 2)	21.09 (4.8)	18.97 (5.6)	**16.13** (1.9)



(a) LMPCC



(b) Dynamic Window

Figure 4.8: Dynamic collision avoidance scenario. The red crosses represent the global path to follow (waypoints). The blue and magenta lines represent the trajectory executed by our LMPCC (top) and by the Dynamic Window (bottom), respectively. The trajectories of the two pedestrians are represented by the green and magenta circles. In the lower case (dynamic window) the robot reacts late and the pedestrians must actively avoid it.

Experimental Results

We use the previous setup to compare the LMPCC and DW methods on a real scenario with two pedestrians. We do not test the CADRL method because the current open-source implementation does not allow static collision avoidance for unconstrained scenarios such as the faculty corridor depicted in Figure 4.8. Figure 4.8 shows one representative run of our method (top) and the DW (bottom). We observe that the proposed method reacts in advance to avoid the pedestrians, resulting in a larger clearance distance. In contrast, the DW reacts late to avoid the pedestrian, which has to avoid the robot himself actively. Our proposed method was able to navigate safely in all of our experiments with static and two pedestrians.

4.5 Results - Autonomous Car

To validate the applicability of our method to more complex robot models, we have performed a simulation and real experiments on an autonomous vehicle.

The planner commands the acceleration and front steering. The car follows a global reference path while staying within the road boundaries (i.e., the obstacle-free region) and avoiding moving obstacles (such as a simulated cyclist proceeding in the direction of the car and a pedestrian crossing the road in front of the car).

The simulation setup uses a 9 DoF non-linear car model (three rigid bodies representing the sprung body, front, and rear axles) developed in MATLAB/SimMechanics [182]. The tire dynamics are modelled using Delft-Tyre 6.2 with a Magic Formula steady-state slip model describing nonlinear slip forces and moments [183]. The car model runs on a Windows PC with an Intel Xeon CPU running at 3.60 GHz. The car model is simulated at 100 Hz, while the localization, perception, and control on the ROS machine (running Ubuntu 18.04.1 LTS) send messages at 25 Hz (as in the setup we have on the real vehicle). The modules used in the experimental setup are implemented on a PC (mounted on board the car) running Ubuntu 18.04.1 LTS with an Intel(R) Core(TM) i7-6900K CPU at 3.20GHz. In addition, the PC contains two Titan X (Pascal) GPUs for stereo matching and Vulnerable Road User (VRU), i.e., pedestrian or cyclist, detection.

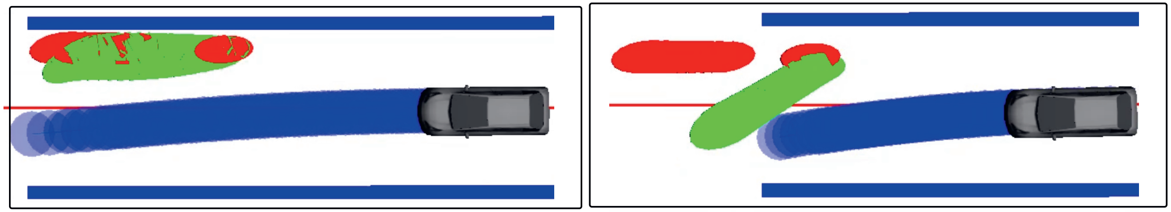
For safety reasons, we tested the interactions of the vehicle with a cyclist and two pedestrians in simulation. Furthermore, we tested the interactions between the real vehicle and a pedestrian dummy during our experiments. Our design was able to adapt the vehicle behavior to different initial configurations (e.g., different reference velocities for the vehicle and different behavior of the pedestrians and cyclist). We have evaluated our method in the following scenarios:

S1 Figure 4.9 shows the simulation results obtained from the interaction with a cyclist and shows the benefits of using the estimated paths of the cyclist in the planner. The cyclist decides to turn at an upcoming intersection. Thanks to the perception module that predicts the path (Figure 4.9a), the car starts to brake (notice that the length of the blue path shrinks) before the cyclist starts to turn (Figure 4.9b) and adapts its path to prevent a possible collision (Figure 4.9c), while remaining within the road boundaries. Without the prediction the cyclist will turn represented in green (e.g., with just a constant velocity model) the car would not have enough time to react to the turning cyclist. Notice that during the maneuver the planner commands the car to brake (reducing its speed) for the safety of the cyclist (second plot from the left in Figure 4.9b). This is possible thanks to the MPCC formulation that, compared to classical path-following approaches, allows the controller more flexibility to determine the state trajectories.

S2 Figures 4.10 shows simulation results with two pedestrians crossing in front of the car. This scenario shows how our vehicle handles multiple VRUs. The vehicle starts to pass at left the first pedestrian (Figure 4.10a). Then, given that the first pedestrian continues to cross the street (from top to bottom) the vehicle plans to pass at right (Figure 4.10b). During the maneuver, the vehicle encounters the second pedestrian (crossing the road from bottom to top), and plans a path to avoid both pedestrians (Figure 4.10c). The two pedestrians cross the road safely and the car returns to its path.

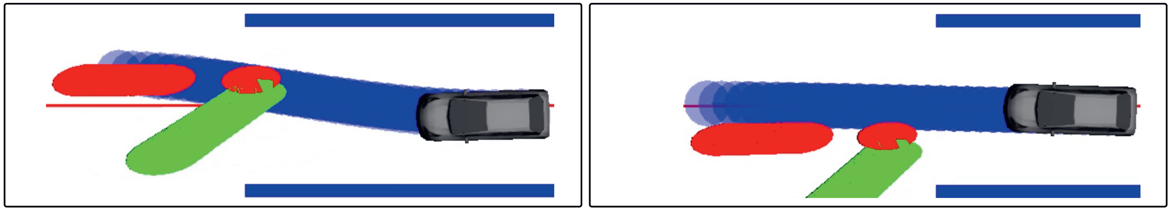
S3 Figures 4.11 and 4.12 show the experimental results. As Figure 4.11 depicts, the vehicle is able to overtake the pedestrian dummy by taking into account its predicted position. At the same time, the car is also able to increase its speed to reach its desired speed (3 m/s), as Figure 4.12 shows. Figure 4.12 shows that the measured vehicle motion, closely follows the desired motion, with some noise in the acceleration and a small delay in steering (approx 0.2 s) caused by physical steering-wheel limitations. Nevertheless, the vehicle is able to safely pass the pedestrian dummy.

In all the cases studied, the LMPCC provided suitable paths for the vehicle to follow to ensure the safety of the VRUs by taking into account their predicted paths (with behavioral uncertainties) provided by the perception module. If sufficient space was available, the vehicle passed the VRUs, planning agile maneuvers when needed (e.g., Scenario *S2*). If passing was unsafe the vehicle reduced its speed or stopped (e.g., Scenario *S1*).



(a) The vehicle plans to overtake the cyclist at the left.

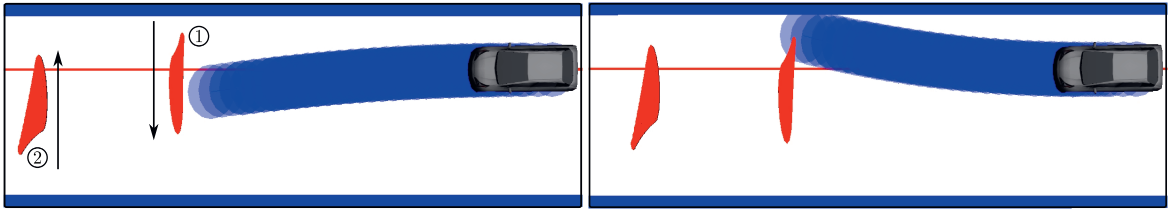
(b) The vehicle brakes based on the cyclist's estimated path.



(c) The vehicle plans based on cyclist's estimated path.

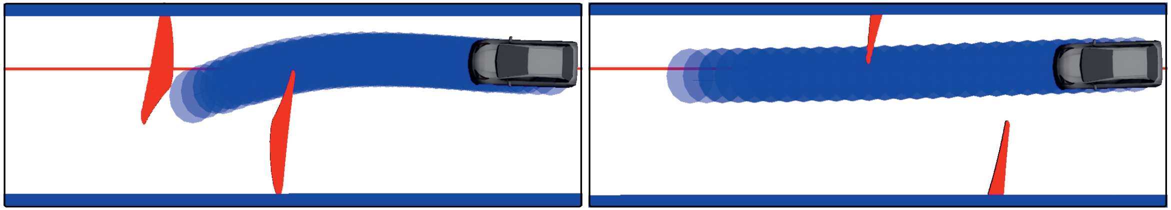
(d) The vehicle returns to its path.

Figure 4.9: S1: turning cyclist. *The vehicle adapts its predicted path based on the two estimated paths of the cyclist. The blue lines are the road boundaries, the red line is the global path, the blue path (circles) is the predicted trajectory of the car, the green and red paths (ellipses) represent the predicted trajectory of the cyclist provided by the perception module (as a mixture of two Gaussians). The red path is associated with the prediction that the cyclist will go straight at the intersection, while the green path is associated with the prediction that the cyclist will turn at the intersection.*



(a) The vehicle plans to pass at left the first pedestrian.

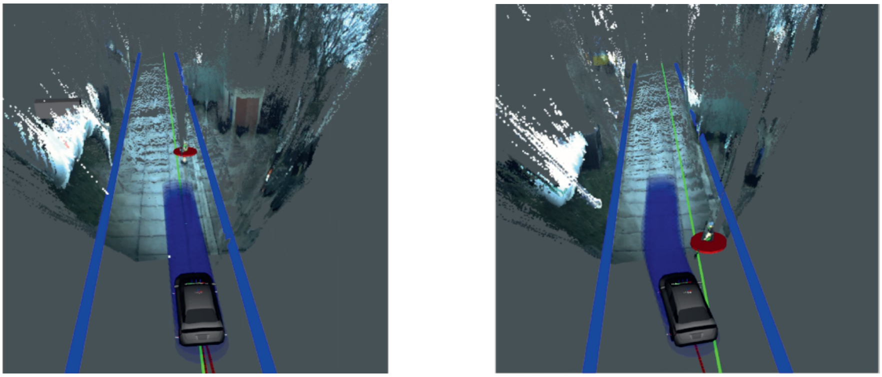
(b) The vehicle plans to pass at right the first pedestrian.



(c) The vehicle encounters the second pedestrian and plans a path to avoid both pedestrians.

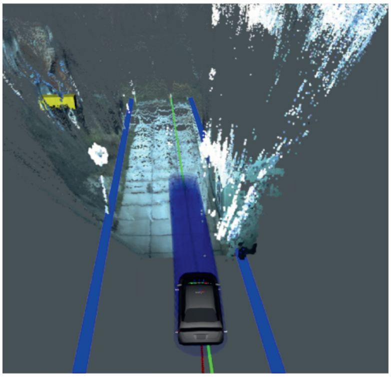
(d) The vehicle returns to its path and the pedestrians safely cross the road.

Figure 4.10: S2: two pedestrians. *The vehicle adapts its predicted path based on the estimated path (red ellipses) of each pedestrian. Notice that the size of each ellipses grows over the horizon due to the uncertainties on the pedestrian positions over time.*



(a) Start of the overtaking.

(b) During the overtaking.



(c) End of the overtaking.

Figure 4.11: S3: experimental results with a pedestrian dummy. *Trajectory of the vehicle during one of our experiments with our research platform. The blue lines depict the road boundaries, the green line is the global path, the blue circles depict the trajectory planned by the local planner, the red ellipses represent the dummy's predicted position.*

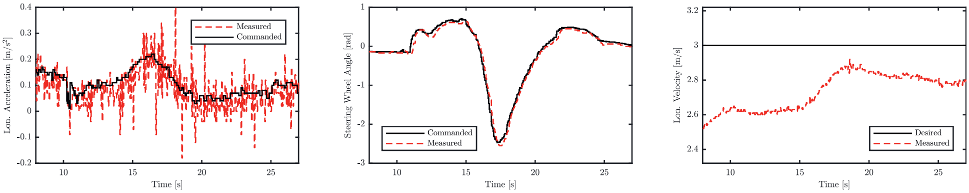


Figure 4.12: S3: experimental results with a pedestrian dummy. *Acceleration, steering wheel angle, and longitudinal velocity of the vehicle.*

4.6 Conclusions and Future Work

This chapter introduced a local planning approach based on Model Predictive Contouring Control (MPCC) to navigate a mobile robot in dynamic, unstructured environments safely. Our local MPCC (LMPCC) relies on an upper bound of the Minkowski sum of a circle and an ellipse to safely avoid dynamic obstacles and a set of convex regions in free space to avoid static obstacles. We compared our design with three baseline approaches (classical MPC, Dynamic Window, and CADRL). The experimental results demonstrate that our method outperforms the baselines in static and dynamic environments. Moreover, the light implementation of our design shows the scalability of our approach up to six agents and allows us to run all algorithms on-board. Finally, we showed the applicability of our design to more complex robots by testing our method on an autonomous car.

5

Social-VRNN: One-Shot Multi-modal Trajectory Prediction for Interacting Pedestrians

---

This chapter is based on:

- B. Brito, H. Zhu, W. Pan and J. Alonso-Mora, "Social-VRNN: One-Shot Multi-modal Trajectory Prediction for Interacting Pedestrians," *Conference on Robot Learning (CoRL)*, 2020

Code: https://github.com/tud-amr/social_vrnn

5.1 Introduction

In the previous Chapter 4, we have developed a motion planning algorithm for autonomous navigation in unstructured, dynamic environments via local model predictive contouring control (LMPCC). The method relies on a simple Constant Velocity (CV) model to predict the dynamic agent trajectories and perform predictive collision avoidance. Nevertheless, CV predictions ignore the interaction between the robot and the other agents and do not consider the environment constraints.

This chapter tackles the prediction problem and proposes a new deep learning model reasoning about the environment, interaction and multi-modality learned from data. Prediction of human motions is key for safe navigation of autonomous robots among humans in cluttered environments. Therefore, autonomous robots, such as service robots or autonomous cars, shall be capable of reasoning about the intentions of pedestrians to accurately forecast their motions. Such abilities will allow planning algorithms to generate safe and socially compliant motion plans [184, 185].

Generally, human motions are inherently uncertain and multi-modal [186]. The uncertainty is caused by partial observation of the pedestrians' states and their stochastic dynamics. The multimodality is due to interaction effects between the pedestrians, the static environment and non-convexity of the problem. For instance, as Fig. 5.1 shows, a pedestrian can decide to either avoid a static obstacle or engage in a non-verbal joint collision-avoidance maneuver with the other upcoming pedestrian, avoiding on the right or left. Hence, to accurately predict human motions, inference models providing multi-modal predictions are required.

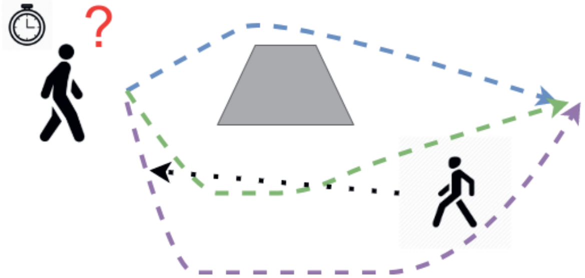


Figure 5.1: Illustration of a scenario where there are multiple ways that two pedestrians can avoid a collision. We present a method that given the same observed past, predicts multiple socially acceptable trajectories in crowded scenes.

A large number of prediction models have been proposed. However, some of these approaches only predict the mean behavior of the agents [85]. Others apply different techniques to model uncertainty such as ensemble modeling [187], dropout during inference [188] or learn a generative model and generate several trajectories by sampling randomly from the latent space [91]. Recently, Generative Adversarial Networks (GANs) have been employed for multi-modal trajectory prediction by randomly sampling the latent space to generate diverse trajectories [93]. Nevertheless, these methods have two main drawbacks. First, GANs are difficult to train and may fail to converge during training. Second, they require a large number of samples to achieve good prediction performance which is impracticable for real-time motion planning. Moreover, these approaches assume an independent prior across different timesteps ignoring the existing time dependencies on trajectory prediction problems.

The goal is to develop a prediction model suitable for interaction-aware autonomous navigation. Hence, we address these limitations with a novel generative model for multi-modal trajectory prediction based on Variational Recurrent Neural Networks (VRNNs) [189]. We treat the multi-modal trajectory prediction problem as modeling the joint probability distribution over sequences.

This chapter's main contribution is a new interaction-aware variational recurrent neural network (Social-VRNN) design for one-shot multi-modal trajectory prediction. By following a variational approach, our method achieves faster convergence in comparison with GAN -based approach. Moreover, employing a time-dependent prior over the latent space enables our model to achieve state-of-the-art performance and generate diverse trajectories with a single network query.

To this end, we propose a training strategy to learn more diverse trajectories in an interpretable fashion. Finally, we present experimental results demonstrating that our method outperforms the state-of-the-art methods on both simulated and real datasets using one-shot predictions.

5.2 Variational Recurrent Neural Network

In this section, we present our Variational Recurrent Neural Network (VRNN) for multi-modal trajectory prediction, depicted in Fig. 5.2.

5.2.1 Multi-modal Trajectory Prediction Problem Formulation

Consider a navigation scenario with n interacting agents (pedestrians) navigating on a plane $\mathcal{W} = \mathbb{R}^2$. The dataset $\mathbf{D}$ contains information about the i-th pedestrian trajectory $\tau^i_{1:N} = \{(\mathbf{p}^i_1, \mathbf{v}^i_1), \dots, (\mathbf{p}^i_N, \mathbf{v}^i_N)\}$ with N utterances and their corresponding surrounding static environment $\mathcal{O}^i_{\text{env}} \subset \mathcal{W}$, for $i \in [0, \dots n]$. $\mathbf{v}^i_t = \{v^i_{x,t}, v^i_{y,t}\}$ is the velocity and $\mathbf{p}^i_t = \{p^i_{x,t}, p^i_{y,t}\}$ is the position of the i-th pedestrian at time t in the world frame. Without loss of generality, $t = 0$ indicates the current time and $t = -1$ the previous time-step. $\mathbf{v}^i_{1:T_H} = (\mathbf{v}^i_1, \dots, \mathbf{v}^i_{T_H})$ represents the future pedestrian velocities over a prediction horizon T_H and $\mathbf{v}^i_{-T_O:0}$ the pedestrian past velocities within an observation time T_O. Throughout this chapter the subscript i denotes the *query-agent*, i.e., the agent that we want to predict its future motion, and $-i$ the collection of all the other agents. Bold symbols are used to represent vectors and the non bold x and y subscripts are used to refer to the x and y direction in the world frame. $\mathbf{x}^i_0 = \{\mathbf{v}^i_{-T_O:0}, \mathbf{p}^{-i}_0, \mathcal{O}^i_{\text{env}}\}$ represents the *query-agent* current state information. To account for the uncertainty and multimodality of the i-th pedestrian's motion, we seek a probabilistic model $f(\theta)$ with parameters θ over a set of M different trajectories $\forall m \in [0, M]$:

$$p(\mathbf{v}^{i,m}_{1:T_H} | \mathbf{x}^i_0) = f(\mathbf{x}^i_0, \theta) \tag{5.1}$$

where m is the trajectory index. The probability is conditional to other agents states and the surrounding environment to model the interaction and environment constraints.

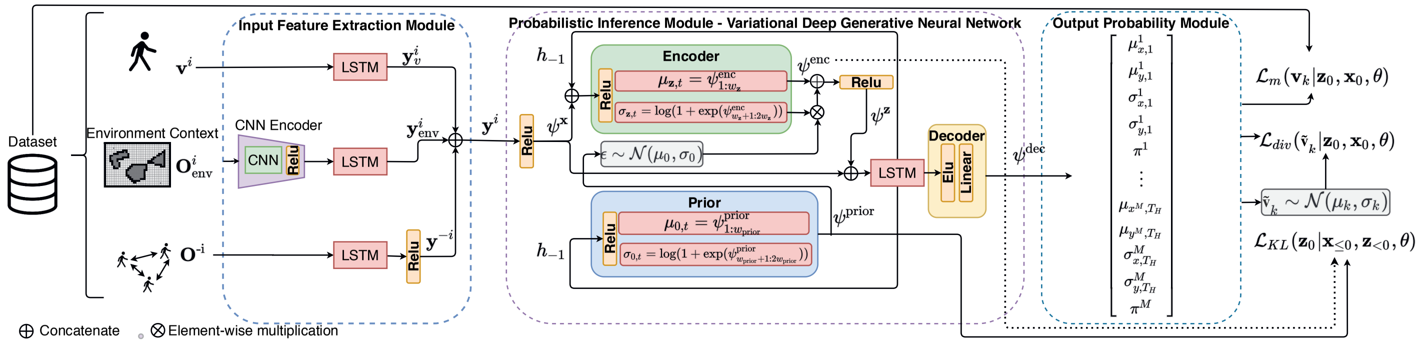

Dataset
Environment Context
$\mathbf{v}^i$
$\mathbf{O}^i_{\text{env}}$
$\mathbf{O}^{\text{-i}}$
Input Feature Extraction Module
CNN Encoder
CNN
Relu
LSTM
LSTM
LSTM
Relu
$\mathbf{y}^i_v$
$\mathbf{y}^i_{\text{env}}$
$\mathbf{y}^{-i}$
$\mathbf{y}^i$
Probabilistic Inference Module - Variational Deep Generative Neural Network
h_{-1}
Relu
$\psi^{\mathbf{x}}$
Encoder
Relu
$\mu_{\mathbf{z},t} = \psi^{\text{enc}}_{1:w_{\mathbf{z}}}$
$\sigma_{\mathbf{z},t} = \log(1 + \exp(\psi^{\text{enc}}_{w_{\mathbf{z}}+1:2w_{\mathbf{z}}}))$
ψ^{enc}
Relu
$\psi^{\mathbf{z}}$
$\epsilon \sim \mathcal{N}(\mu_0, \sigma_0)$
Prior
Relu
$\mu_{0,t} = \psi^{\text{prior}}_{1:w_{\text{prior}}}$
$\sigma_{0,t} = \log(1 + \exp(\psi^{\text{prior}}_{w_{\text{prior}}+1:2w_{\text{prior}}}))$
ψ^{prior}
h_{-1}
LSTM
Decoder
Elu
Linear
ψ^{dec}
Output Probability Module
$\mu^1_{x,1}$
$\mu^1_{y,1}$
$\sigma^1_{x,1}$
$\sigma^1_{y,1}$
π^1
$\vdots$
μ_{x^M,T_H}
μ_{y^M,T_H}
σ^M_{x,T_H}
σ^M_{y,T_H}
π^M
$\mathcal{L}_m(\mathbf{v}_k|\mathbf{z}_0, \mathbf{x}_0, \theta)$
$\mathcal{L}_{div}(\tilde{\mathbf{v}}_k|\mathbf{z}_0, \mathbf{x}_0, \theta)$
$\tilde{\mathbf{v}}_k \sim \mathcal{N}(\mu_k, \sigma_k)$
$\mathcal{L}_{KL}(\mathbf{z}_0|\mathbf{x}_{\leq 0}, \mathbf{z}_{<0}, \theta)$
⊕ Concatenate
⊗ Element-wise multiplication

Figure 5.2: Social-VRNN architecture for multi-modal trajectory prediction composed by: an input feature extraction, a probabilistic inference and output probability module. The first creates a joint representation of the input data $\mathbf{y}^i = \{\mathbf{y}^i_{\mathbf{v}}, \mathbf{y}^i_{env}, \mathbf{y}^{-i}\}$*. The probabilistic inference module (Section 5.2.3) is based on the VRNN [189] incorporating: a encoder network to approximate a time-dependent posterior distribution* $q(\mathbf{z}_0|\mathbf{x}_{\leq 0}, \mathbf{z}_{<0}) \sim \mathcal{N}(\mu_{\mathbf{z},0}, diag(\sigma^2_{\mathbf{z},0}))$ *with* $[\mu_{\mathbf{z},0}, \sigma_{\mathbf{z},0}] = \psi^{enc}(\psi^{\mathbf{x}}(\mathbf{x}_0), \mathbf{h}_{-1}, \theta_q)$ *with* θ_q *as the approximate posterior model parameters; a decoder network to model the conditional generation distribution* $\mathbf{v}_k|\mathbf{x}_0, \mathbf{z}_0 \sim \mathcal{N}(\mu_{\mathbf{x},0}, diag(\sigma^2_{\mathbf{x},0}))$ *with* $[\mu_{\mathbf{v},1:T_H}, \sigma_{\mathbf{v},1:T_H}] = \psi^{dec}(\psi^{\mathbf{z}}(\mathbf{z}_0), \psi^{\mathbf{x}}_t(\mathbf{x}_0), \mathbf{h}_{-1}, \theta_{\text{dec}})$ *with* θ_{dec} *as the inference model parameters; a prior on the latent random variable* $\mathbf{z} \sim \mathcal{N}(\mu_{prior,0}, \sigma_{prior,0})$ *conditional to the hidden-state of the decoder network* $[\mu_{prior,0}, \sigma_{prior,0}] = \psi^{prior}(\mathbf{h}_{-1}, \theta_{prior})$ *with parameters* θ_{prior}*. Finally, the output probability module is a GMM (Section 5.2.4).*

5.2.2 Input Feature Extraction Module

This module creates a joint representation of three sources of information: the query-agent state, the environment context and social context. The first input is a sequence of T_O history velocities $\mathbf{v}^i_{-T_O:0}$ of the query-agent. The second input is a local occupancy grid O^i_{env}, centered at the query-agent containing information about static obstacles (environment context) with width D_x and height D_y. Here, we use the global map provided with publicly available datasets [90, 190]. In a real scenario, the map information can be obtained by building a map offline [191] or local map from online [192] using onboard sensors such as Lidar. Due to its high dimensionality, a convolution neural network (CNN) is used to obtain a compressed representation of this occupancy map while maintaining the spatial context. The encoder parameters are obtained by pre-training an Encoder-Decoder structure to minimize $\mathcal{L}_{\text{env}} = \sum_{i=1}^{D_x} \sum_{j=1}^{D_y} (\hat{\mathbf{O}^i_{\text{env}}} - \mathbf{O}^i_{\text{env}})^2$, as proposed in [85]. In addition, an LSTM layer is added to the first two input channels, modeling the existing time-dependencies.

The third input provides information about the interaction among the pedestrians containing information about their relative dynamics and spatial configuration. More specifically, it is a vector $\mathbf{O}_0^{-i} = [\mathbf{p}_0^{-1} - \mathbf{p}_0^i, \mathbf{v}_0^{-1} - \mathbf{v}_0^i, \dots \mathbf{p}_0^{-n} - \mathbf{p}_0^i, \mathbf{v}_0^{-n} - \mathbf{v}_0^i]$ with the positions and velocities of the surrounding pedestrians relative to the query-agent. This input vector is then fed into an LSTM, allowing to create a fixed-size representation of the query's agent social context and to consider a variable number of surrounding pedestrians. Finally, the outputs of each channel are concatenated creating a compressed and time-dependent representation of the input data $\mathbf{y}^i = \{\mathbf{y}^i_{\mathbf{v}}, \mathbf{y}^i_{\text{env}}, \mathbf{y}^{-i}\}$. Note that we only use past information about the query-agent velocities. For the other inputs only the current information is used.

5.2.3 Probabilistic Inference Module

The probabilistic inference module is based on the structure of the VRNN, as depicted in Fig. 5.2. It contains three main components: a prior model, a encoder model and decoder model. We use a fully connected layer (FCL) with Relu activation as the encoder model ψ^{enc}, the feature extractor of the joint input $\psi^{\mathbf{x}}$ and of the latent random variables $\psi^{\mathbf{z}}$, and the representation of the prior distribution ψ^{prior}. $\{\theta_{\text{enc}}, \theta_{\mathbf{x}}, \theta_{\mathbf{z}}, \theta_{\text{prior}}\}$ are the network parameters of $\{\psi^{\text{enc}}, \psi^{\mathbf{x}}, \psi^{\mathbf{z}}, \psi^{\text{prior}}\}$, respectively. The output vectors $\{\psi^{\text{enc}}_{\tau}, \psi^{\text{prior}}\}$ are then used to model the approximate posterior and prior distribution. We split the output vectors into two parts to model the mean and variance, as represented in Fig. 5.2, and apply the following transformations to ensure a valid predicted distribution: $[\mu_{\text{prior}}, \mu_{\mathbf{z}}] = [\psi^{\text{prior}}_{1:w_{\text{prior}}}, \psi^{\text{enc}}_{1:w_{\mathbf{z}}}]$ and $[\sigma_{\text{prior}}, \sigma_{\mathbf{z}}] = [\exp \psi^{\text{prior}}_{w_{\text{prior}}:2w_{\text{prior}}}, \exp \psi^{\mathbf{z}}_{w_{\mathbf{z}}:2w_{\mathbf{z}}}]$.

$2w_{\text{prior}}$ and $2w_{\mathbf{z}}$ are the output vector size of the prior and latent random variable, respectively. This ensures that the standard deviation is always positive. Furthermore, we employ a LSTM layer as the RNN model propagating the hidden-state for the prior model and encoding the time-dependencies for the generative model. In contrast to [189] our generation model conditionally depends on the previous inputs:

$$\begin{aligned} \mathbf{v}_k | \mathbf{x}_0, \mathbf{z}_0 &\sim \mathcal{N}(\mu_{\mathbf{v},k}, \text{diag}(\sigma^2_{\mathbf{v},k})) \\ [\mu_{\mathbf{v},k}, \sigma_{\mathbf{v},k}] &= \psi^{\text{dec}}(\psi^{\mathbf{z}}(\mathbf{z}_0), \psi^{\mathbf{x}}(\mathbf{y}^i_0), \mathbf{h}_{-1}) \end{aligned} \tag{5.2}$$

Lastly, the decoder model consists of two FC layers, with ELU [193] and linear activation, directly connected to the output of the LSTM network. Our models outputs in one shot T_H steps considering the compressed and time-dependent input representation $\mathbf{y}_0^i$.

5.2.4 Multi-modal Trajectory Prediction Distribution

To predict one-shot multi-modal trajectories,we model the output of our network as a Gaussian Mixture Model (GMM), similar to [194] and [195], with $M > 1$ modes accounting for the multimodality of the pedestrian's motion. For each mode $m \in \{1, \dots, M\}$, we predict a sequence of future pedestrian velocities $v_{1:T_H}^{i,m}$ represented by a bivariate Gaussian $v_k^{i,m} \sim \mathcal{N}(\mu_{x,k}^{i,m}, \mu_{y,k}^{i,m}, \sigma_{x,k}^{i,m}, \sigma_{y,k}^{i,m})$, $k = 1, 2, \dots, T_H$, capturing its motion uncertainty. Consequently, a modal trajectory is defined as a sequence of independent bivariate Gaussian's with length T_H. The M modes represent a set of M possible trajectories resulting in the following probabilistic model:

$$p(\mathbf{v}_k^i|\mathbf{x}_0, \mathbf{z}_0, \mathbf{h}_0, \theta) = \sum_{m=1}^{M} \pi_m p_G(\mu_{k,m}^i, \sigma_{k,m}^i) \tag{5.3}$$

where p_G is the probability density function of multivariate Gaussian distributions, $\theta = \{\theta_{\text{enc}}, \theta_{\text{dec}}, \theta_{\mathbf{x}}, \theta_{\mathbf{z}}, \theta_{\text{prior}}\}$ are the model parameters, $\mu_k^{i,m} = [\mu_{x,k}^{i,m}, \mu_{y,k}^{i,m}]$ and $\sigma_k^{i,m} = [\sigma_{x,k}^{i,m}, \sigma_{y,k}^{i,m}]$ are the mean and standard deviation of the predicted velocity vectors for the m-th predicted trajectory with likelihood π_m at time-step k, respectively. The transformations described in Sec.5.2.3 and Fig.5.2 are applied to the network outputs ψ_τ^{dec} to ensure a valid distribution parametrization.

5.2.5 Improving Diversity

Generative models have the key advantage of allowing to perform inference by randomly sampling the latent random variable $\mathbf{z}$ from some prior distribution. Here, we propose a strategy to induce our model to learn a more “diverse" distribution of trajectories in a interpretable fashion, similar to [196]. Our VRNN models a generative distribution conditionally dependent on the input representation vector $\mathbf{y}^{\mathbf{i}}$, which is composed by three sub-vectors $\{\mathbf{y}_{\text{v}}^i, \mathbf{y}_{\text{env}}^i, \mathbf{y}^{-i}\}$. Now, let's assume that each input vector is a random variable with the following distribution:

$$\mathbf{y}_{\text{v}}^i \sim \mathcal{N}(\mathbf{y}_{0,\text{v}}^i, \sigma_{\text{v}}) \tag{5.4}$$

$$\mathbf{y}_{\text{env}}^i \sim \mathcal{N}(\mathbf{y}_{0,\text{env}}^i, \sigma_{\text{env}}) \tag{5.5}$$

$$\mathbf{y}^{-i} \sim \mathcal{N}(\mathbf{y}_0^{-i}, \sigma_{-i}) \tag{5.6}$$

where $\{\mathbf{y}_{\text{v}}^i, \mathbf{y}_{\text{env}}^i, \mathbf{y}^{-i}\}$ are random variables representing the variability of the agent state, the environment and surrounding agents context, respectively. $\{\sigma_{\text{v}}, \sigma_{\text{env}}, \sigma_{-i}\}$ are the variance of each input channel and are considered as hyperparameters of our model. Hence, by sampling from these input distributions we can condition the generation distribution of $\mathbf{x}$ according with the uncertainty on the pedestrian state or the environment context and generate different trajectories $\tilde{\mathbf{v}}^i_{1:T_H}$ by varying the pedestrian conditions. Then, we introduce a loss function which motivates our model to cover the generated trajectories as the following cross-entropy term:

$$\mathcal{L}_{div} = \sum_{m=1}^{M}\sum_{k=1}^{T_H} -\mathbb{E}[\log p_G(\tilde{\mathbf{v}}_k|\mathbf{x}_0, \mathbf{z}_0)] \tag{5.7}$$

where $\tilde{\mathbf{v}}_{m,k}$ is a velocity sample at time-step k from the m-th sampled trajectory.

5.2.6 Training Procedure

The model is trained end-to-end except for the CNN which is pre-trained. We train it using back-propagation through time (BTTP) with fixed truncation depth t_{trunc}. Furthermore, we apply the reparametrization trick [26] to obtain a continuous differentiable sampler and train the network using backpropagation. We learn the data distribution by minimizing a timestep-wise variational lower bound with annealing KL-Divergence as loss function [197]:

$$\mathcal{L} = \mathcal{L}_m + \lambda * (\mathcal{L}_{KL} + \mathcal{L}_{div}) \tag{5.8a}$$

$$\mathcal{L}_m = \sum_{m=1}^{M}\sum_{k=1}^{T_H} -\mathbb{E}_{\mathbf{x}_0 \sim \mathbf{D}}[\log \pi_m p_G(\mathbf{v}_k|\mathbf{z}_0, \mathbf{x}_0)] \tag{5.8b}$$

$$\mathcal{L}_{KL}(\mathbf{z}_0|\mathbf{x}_{\leq 0}, \mathbf{z}_{<0}) = \lambda \mathrm{KL}(q(\mathbf{z}_0|\mathbf{x}_{\leq 0}, \mathbf{z}_{<0}) || p_G(\mathbf{z}_0|\mathbf{x}_{<0}, \mathbf{z}_{<0})) \tag{5.8c}$$

where λ is the annealing coefficient. The first term represents the reconstruction loss (Eq. 5.8b) and the second the KL-Divergence between the approximated posterior $q(\mathbf{z}_0|\mathbf{x}_{\leq 0}, \mathbf{z}_{<0})$ (Eq. 5.8c) and the prior distribution of $\mathbf{z}$. Here, the prior over the latent random variable $\mathbf{z}$ is chosen to be a simple Gaussian distribution with mean and variance $[\mu_{\text{prior},0}, \sigma_{\text{prior},0}] = \psi^{\text{prior}}(\mathbf{h}_{-1})$ depending on the previous hidden state. During training we aim to find the model parameters which minimize the loss function presented in Equation 5.8a. The annealing coefficient allows the model first to learn the parameters that fit the data well and later in the training phase to match the prior distribution and improve the diversity of the predicted trajectories.

5.3 Experiments

In this section, we show the obtained results of our generative model for simulation and real data. We present a qualitative analysis and performance results of our method against three baselines. To evaluate the performance of our model against the proposed baselines we use the following evaluation metrics: the average displacement error (ADE) and the final displacement error (FDE). The first two assess the prediction performance. For the models outputting probability distributions, the mean values are used to compute the ADE and FDE metrics. For the multi-modal distributions, we use the trajectory with the minimum error as in [93].

5.3.1 Experimental Settings

We trained our model using RMSProp [198] which is known to perform well in non-stationary problems with a initial learning rate $\alpha = 10^{-4}$ exponentially decaying at a rate of 0.9 and a mini-batch size of 16. We used a KL annealing coefficient $\lambda = \tanh(\frac{\text{step}-10^4}{10^3})$, with step as the training step. We set the diversity weight β to 0.2 and $\{\sigma_{\mathbf{x}^v}, \sigma_{\mathbf{x}^{\text{env}}}, \sigma_{\mathbf{x}^{-i}}\} = \{0.2, 0.2, 0\}$. Additionally, to avoid gradient explosion we clip the gradients to 1.0. We trained and evaluated our model for different prior, latent random variable and input feature vector sizes. The configuration achieving lower validation error was $\{128, 128, 512\}$, for the prior, latent random variable and input feature vector size, respectively. Moreover, we use $M = 3$ mixture components for the models using a GMM as the output function. We set $T_H = 12$ prediction steps corresponding to 4.8 s of prediction horizon and $T_O = 8$ as used in previous methods [92, 93]. The models were implemented using Tensorflow [199] and were trained on a NVIDIA GeForce GTX 980 requiring 2×10^4 training steps, or approximately 2 hours. The simulation datasets were obtained with the open-source ROS implementation of the Social Forces model [80]. Our VRNN will be released open source.

5.3.2 Performance Evaluation

We compared our model with the following state-of-art prediction baselines:

- *LSTM-D* [85]: A deterministic interaction-aware model, incorporating the interaction between the agents and static obstacles.
- *SoPhie* [92]: a GAN model implementing a Social and Physical attention mechanism.
- *Social-ways* (S-Ways) [93]: The state-of-art GAN based method for multi-modal trajectory prediction.
- *STORN* [200]: Our VRNN model considering a time-independent prior as a Gaussian distribution with zero mean and unit variance.

Table 5.1: Performance results of our proposed method (Social-VRNN) vs. baselines. The results presented for the Social Ways with 30 samples ($K = 30$) and SoPhie method were taken from [86] and [92], respectively. The ADE and FDE values are separated by slash. The average values (AVG) only consider the results for the real datasets. The results for using three samples ($K = 3$) of S-Ways were obtained from the open-source implementation provided by [93].

	Deterministic	Stochastic				
	Single Sample	Multiple Samples			Single Sample	
Dataset	LSTM	SoPhie	S-Ways ($K = 30$)	S-Ways ($K = 3$)	STORN	Social-VRNN
ETH	0.40 / 0.65	0.70 / 1.43	**0.39 / 0.64**	0.78 / 1.48	0.73 / 1.49	**0.39** / 0.70
Hotel	0.45 / 0.75	0.76 / 1.67	0.39 / 0.64	0.53 / 0.95	1.33 / 1.45	**0.35 / 0.47**
Univ	1.02 / 1.54	0.54 / 1.24	0.55 / 1.31	0.81 / 1.53	0.82 / 1.17	**0.53 / 0.65**
ZARA01	0.35 / 0.68	**0.30 / 0.63**	0.44 / 0.64	0.87 / 1.30	0.91 / 1.52	0.41 / 0.70
ZARA02	0.54 / 0.92	**0.38** / 0.78	0.51 / 0.92	1.27 / 2.13	0.91 / 1.52	0.51 / **0.55**
AVG	0.55 / 0.90	0.54 / 1.15	0.46 / 0.83	0.86 / 1.47	0.94 / 1.43	**0.44 / 0.61**

We use the open-source implementation of [93] to obtain the results for S-Ways considering only 3 samples ($K = 3$) as the number of trajectories predicted by our method and as suggested in [201]. We adopt the same dataset split setting as in [93] using 4 sets for training and the remaining set for testing. Aggregated results in Table 5.1 show that our method outperformed the deterministic baselines, STORN, and S-Ways using three samples. Moreover, the results show that our method achieves comparable performance with state-of-the-art methods using a high number of samples on the Zara01, Zara02 and ETH datasets. In contrast, our method achieves the best performance on the Hotel and Univ datasets. Finally, the poor performance of the STORN model results show that employing a time-dependent prior improves the prediction performance significantly.

5.3.3 Qualitative Analysis

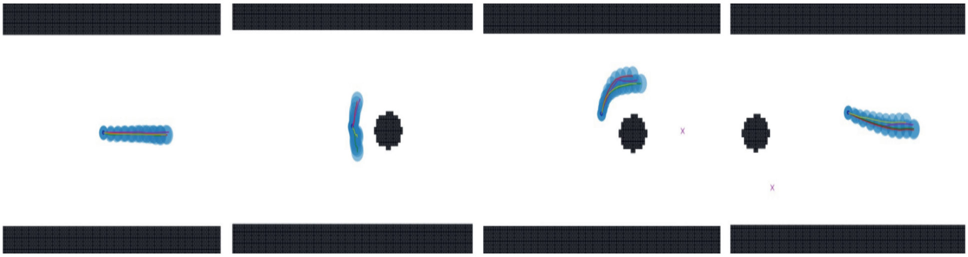
In this section we present prediction results for simulated and real scenarios, as depicted in Fig. 5.3. We have created two datasets to demonstrate this multi-modal behavior with static obstacles (Fig.5.3(a)), and other pedestrians (Fig. 5.3(b)). Figure 5.3(a) shows the ability of our method to predict different trajectories according to the environment structure. Figure 5.3(b) demonstrates that our method can scale to more complex environments, with several pedestrians and obstacles, and predict different motion hypotheses.

Moreover, we evaluate our method on real data using the publicly available datasets [90, 190].

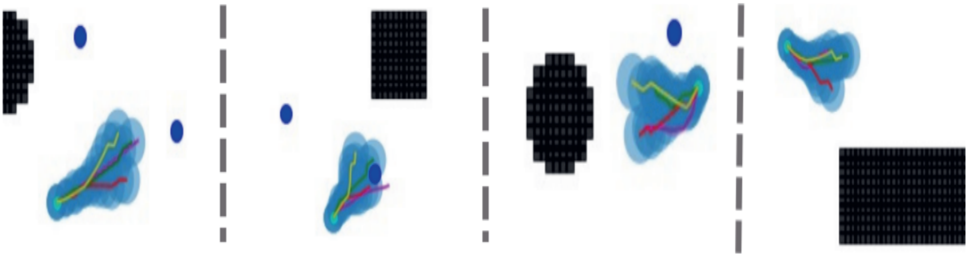
In Fig. 5.3(c) on the left, our model infers two possible trajectories for the pedestrian to avoid a tree. In addition, in the central and right images of Fig. 5.3(c), our model predicts two possible trajectories to move through the crowd. Finally, Fig. 5.4 shows predicted trajectories for both Social-VRNN and Social-Ways model in a crowded scene. The predicted trajectories from the Social-VRNN model can capture two distinct trajectories through the crowd. In contrast, Social-Ways only captures one mode, even considering 30 samples from the baseline model. The presented results demonstrate that our model can effectively infer different trajectories according to the environment and social constraints from a single query. We refer the reader to the video[1] accompanying this chapter for more details on the presented results.

---

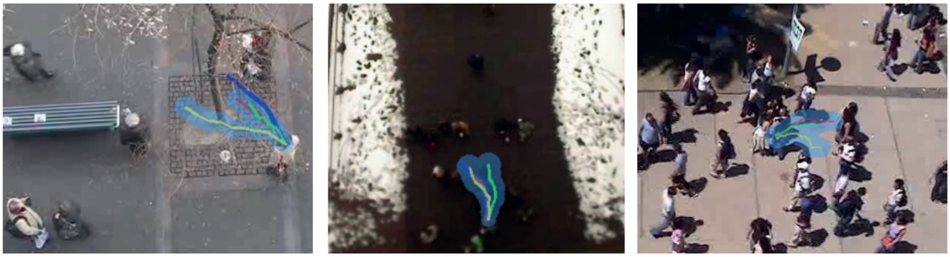
[1] https://youtu.be/tBr5v7TXyG0



a) In this scenario, one agent is moving along a corridor with an obstacle in the middle. The agent is moving from the left to the right. When she finds the obstacle in the middle of its path, our model successfully predicts two hypotheses: going left or right. Once she is already avoiding the obstacle through the left side, the model predicts three hypotheses for the pedestrian to continue its collision avoidance maneuver, with varying clearance levels. Finally, when she is in free space all the predicted trajectories collapse to a single-mode.



b) This sub-figure illustrates four sample results obtained in a more complex simulated scenario, with several static obstacles and 15 agents. The two left figures show two situations where the agent can avoid another agent on its left, right or by simply move straight because the other will keep moving away. The two right figures show the ability of our model to predict different trajectories that an agent may follow to avoid a static obstacle.



c) Three examples of multi-modal trajectory prediction using our model in real scenarios. In blue is depicted the ground truth trajectory, in red, green and yellow the three possible predicted trajectories, in light blue the one sigma boundary of the predicted trajectory.

Figure 5.3: The scenarios depicted in Fig.5.3(a) and (b) were simulated by using the Social Forces model [80] for the pedestrians. In magenta the real trajectory, in red, green and yellow the mean values of each trajectory hypothesis and, in blue the 1-σ uncertainty boundaries of each trajectory. The dark blue dots represent the other agents. The plotted trajectories correspond to a single network query.

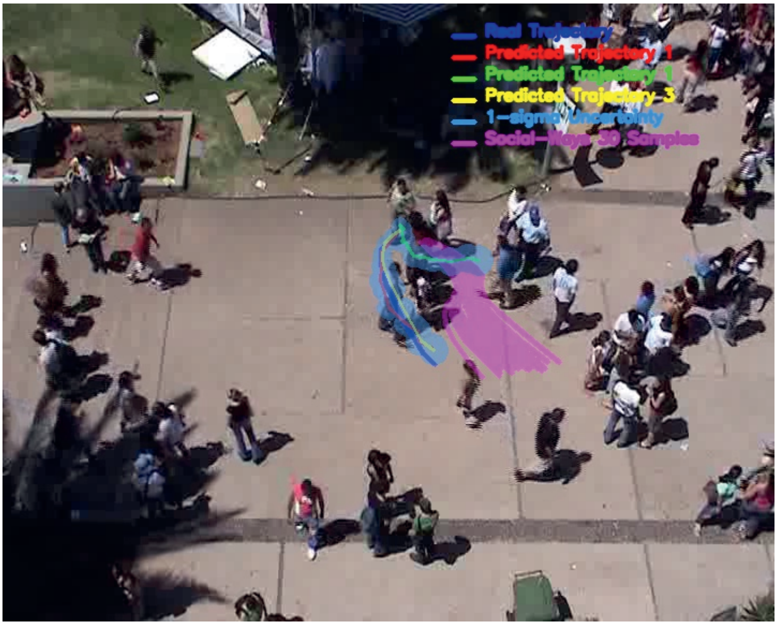


Figure 5.4: Social-VRNN predicted trajectories vs a multi-modal prediction baseline, Social-Ways [93]. In blue is depicted the ground truth trajectory, in red, green and yellow the three possible predicted trajectories by our model, in light blue the one sigma boundary of the predicted trajectory and, in magenta 30 sampled predicted trajectories by the Social-Ways model.

5.4 Conclusions

This chapter introduced a Variational Recurrent Neural Network (VRNN) architecture for multi-modal trajectory prediction in one-shot and considering the pedestrian dynamics, interactions among pedestrians and static obstacles. Building on a variational approach and learning a mixture Gaussian model enables our model to generate distinct trajectories accounting for the static obstacles and the surrounding pedestrians. Our approach allows us to improve the state-of-the-art prediction performance in scenarios with a large number of agents (e.g., Univ dataset) or containing static obstacles (e.g., Hotel dataset) from a single prediction shot. Furthermore, the proposed approach reduces significantly the number of samples needed to achieve good prediction with high accuracy. Future work can integrate the proposed method with a real-time motion planner on a mobile platform for autonomous navigation among pedestrians.

6

Where to go next: Learning a Subgoal Recommendation Policy for Navigation Among Pedestrians

---

This chapter is based on:

- B. Brito, M. Everett, M. How and J. Alonso-Mora, "Where to go next: Learning a Subgoal Recommendation Policy for Navigation in Dynamic Environments." IEEE Robotics and Automation Letters 6.3 (2021): 4616-4623.

Code: https://github.com/tud-amr/go-mpc

6.1 Introduction

Autonomous robot navigation in crowds remains difficult due to the interaction effects among navigating agents. Unlike multi-robot environments, robots operating among pedestrians require decentralized algorithms that can handle a mixture of other agents' behaviors without depending on explicit communication between agents.

Several state-of-the-art collision avoidance methods employ model-predictive control (MPC) with online optimization to compute motion plans that are guaranteed to respect important constraints [7], for instance the LMPCC presented in Chapter 4. These constraints could include the robot's nonlinear kino-dynamics model or collision avoidance of static obstacles and other dynamic, decision-making agents (e.g., pedestrians). Although modern solvers enable real-time motion planning in many situations of interest [202], online optimization becomes less computationally practical for extremely dense scenarios.

A key challenge is that the robot's global goal is often located far beyond the planning horizon, meaning that a local subgoal or cost-to-go heuristic must be specified instead. This is straightforward in a static environment (e.g., using euclidean/diffusion [102] distance), but the presence interactive agents makes it difficult to quantify which subgoals will lead to the global goal quickest. A body of work addresses this challenge with deep reinforcement learning (RL), in which agents learn a model of the long-term cost of actions in an offline training phase (usually in simulation) [17, 18, 111, 113]. The learned model is fast-to-query during online execution, but the way learned costs/policies have been used to date does not provide guarantees on collision avoidance or feasibility with respect to the robot dynamics.

In this chapter, we introduce Goal Oriented Model Predictive Control (GO-MPC), which enhances state-of-art online optimization-based planners with a learned global guidance policy. In an offline RL training phase, an agent learns a policy that uses the current world configuration (the states of the robot and other agents, and a global goal) to recommend a local subgoal for the MPC, as depicted in Figure 6.1. Then, the MPC generates control commands ensuring that the robot and collision avoidance constraints are satisfied (if a feasible solution is found) while making progress towards the suggested subgoal. Our approach maintains the kino-dynamic feasibility and collision avoidance guarantees inherent in an MPC formulation, while improving the average time-to-goal and success rate by leveraging past experience in crowded situations.

The main contributions of this chapter are:

- A goal-oriented Model Predictive Control method (GO-MPC) for navigation among interacting agents, which utilizes a learned global guidance policy (recommended subgoal) in the cost function and ensures that dynamic feasibility and collision avoidance constraints are satisfied when a feasible solution to the optimization problem is found;
- An algorithm to train an RL agent jointly with an optimization-based controller in mixed environments, which is directly applicable to real-hardware, reducing the sim to real gap.

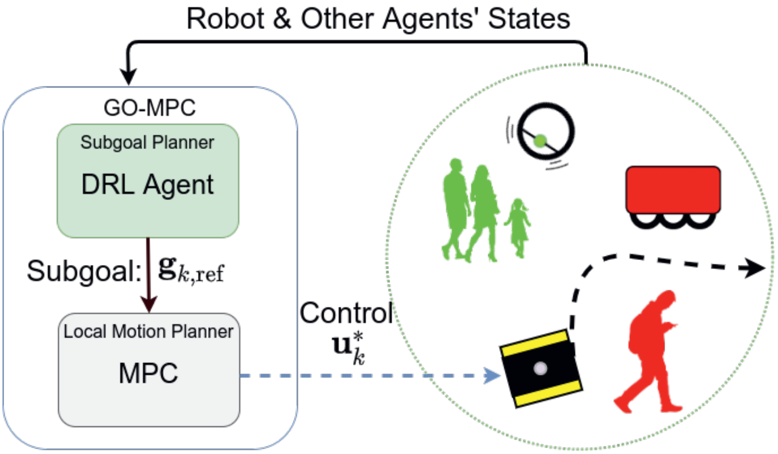


Figure 6.1: Proposed navigation architecture. The subgoal planner observes the environment and suggests the next subgoal position to the local motion planner, the MPC. The MPC then computes a local trajectory and the robot executes the next optimal control command, which minimizes the distance to the provided position reference while respecting collision and kinodynamic constraints.

Finally, we present simulation results demonstrating an improvement over several state-of-art methods in challenging scenarios with realistic robot dynamics and a mixture of cooperative and non-cooperative neighboring agents. Our approach shows different navigation behaviors: navigating through the crowd when interacting with cooperative agents, avoiding congestion areas when non-cooperative agents are present and enabling communication-free decentralized multi-robot collision avoidance.

6.2 Preliminaries

6.2.1 Problem Formulation

Consider a scenario where a robot must navigate from an initial position $\mathbf{p}_0$ to a goal position $\mathbf{g}$ on the plane $\mathbb{R}^2$, surrounded by n non-communicating agents. At each time-step t, the robot first observes its state $\mathbf{s}_t$ (defined in Sec.6.3.1) and the set of the other agents states $\mathrm{S}_t = \bigcup_{i\in\{1,\dots,n\}} \mathbf{s}_t^i$, then takes action $\mathbf{a}_t$, leading to the immediate reward $R(\mathbf{s}_t, \mathbf{a}_t)$ and next state $\mathbf{s}_{t+1} = h(\mathbf{s}_t, \mathbf{a}_t)$, under the transition model h.

We use the superscript $i \in \{1,\dots,n\}$ to denote the i-th nearby agent and omit the superscript when referring to the robot. For each agent $i \in \{0,n\}$, $\mathbf{p}_t^i \in \mathbb{R}^2$ denotes its position, $\mathbf{v}_t^i \in \mathbb{R}^2$ its velocity at step t relative to a inertial frame, and r_i the agent radius. We assume that each agent's current position and velocity are observed (e.g., with on-board sensing) while other agents' motion intentions (e.g., goal positions) are unknown. Finally, $\mathcal{O}_t$ denotes the area occupied by the robot and $\mathcal{O}_t^i$ by each surrounding agent, at time-step t.

The goal is to learn a policy π for the robot that minimizes time to goal while ensuring collision-free motions, defined as:

$$
\begin{aligned}
\pi^* = \underset{\pi}{\operatorname{argmax}} \quad & \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(\mathbf{s}_t, \pi(\mathbf{s}_t, \mathbf{S}_t))\right] \\
\text{s.t.} \quad & \mathbf{x}_{t+1} = f(\mathbf{x}_t, \boldsymbol{u}_t), && \text{(6.1a)} \\
& s_T = \mathbf{g}, && \text{(6.1b)} \\
& \mathcal{O}_t(\mathbf{x}_t) \cap \mathcal{O}_t^i = \emptyset && \text{(6.1c)} \\
& \mathbf{u}_t \in \mathcal{U},\ \mathbf{s}_t \in \mathcal{S},\ \mathbf{x}_t \in \mathcal{X}, && \text{(6.1d)} \\
& \forall t \in [0, T], \quad \forall i \in \{1, \dots, n\}
\end{aligned}
$$

where (6.1a) are the transition dynamic constraints considering the dynamic model f, (6.1b) the terminal constraints, (6.1c) the collision avoidance constraints and $\mathcal{S}$, $\mathcal{U}$ and $\mathcal{X}$ are the set of admissible states, inputs (e.g., to limit the robot's maximum speed) and the set of admissible control states, respectively. Note that we only constraint the control states of the robot. Moreover, we assume other agents have various behaviors (e.g., cooperative or non-cooperative): each agent samples a policy from a closed set $\mathcal{P} = \{\pi_1, \dots, \pi_m\}$ (defined in Sec.6.2.3) at the beginning of each episode.

6.2.2 Agent Dynamics

Real robotic systems' inertia imposes limits on linear and angular acceleration. Thus, we assume a second-order unicycle model for the robot [203]:

$$
\begin{aligned}
&\dot{x} = v\cos\psi \quad && \dot{v} = u_a \\
&\dot{y} = v\sin\psi \quad && \dot{\omega} = u_\alpha \\
&\dot{\psi} = \omega
\end{aligned}
\tag{6.2}
$$

where x and y are the agent position coordinates and ψ is the heading angle in a global frame. v is the agent forward velocity, ω denotes the angular velocity and, u_a the linear and u_α angular acceleration, respectively.

6.2.3 Modeling Other Agents' Behaviors

In a real scenario, agents may follow different policies and show different levels of cooperation. Hence, in contrast to previous approaches, we do not consider all the agents to follow the same policy [17, 204]. At the beginning of an episode, each non-ego agent either follows a cooperative or a non-cooperative policy. For the cooperative policy, we employ the Reciprocal Velocity Obstacle (RVO) [205] model with a random cooperation coefficient[1] $c^i \sim \mathcal{N}(0.1, 1)$ sampled at the beginning of the episode. The "reciprocal" in RVO means that all agents follow the same policy and use the cooperation coefficient to split the collision avoidance effort among the agents (e.g., a coefficient of 0.5 means that

---

[1] This coefficient is denoted as α_A^B in [103]

each agent will apply half of the effort to avoid the other). For the non-cooperative agents, we consider both constant velocity (CV) and non-CV policies. The agents following a CV model drive straight in the direction of their goal position with constant velocity. The agents following a non-CV policy either move in sinusoids towards their final goal position or circular motion around their initial position.

6.3 Method

Learning a sequence of intermediate goal states that lead an agent toward a final goal destination can be formulated as a single-agent sequential decision making problem. Because parts of the environment can be difficult to model explicitly, the problem can be solved with a reinforcement learning framework. Hence, we propose a two-level planning architecture, as depicted in Figure 6.1, consisting of a subgoal recommender (Section 6.3.1) and an optimization-based motion planner (Section 6.2.3). We start by defining the RL framework and our's policy architecture (Section 6.3.1). Then, we formulate the MPC to execute the policy's actions and ensure local collision avoidance (Section 6.3.2).

6.3.1 Learning a Subgoal Recommender Policy

We aim to develop a decision-making algorithm to provide an estimate of the cost-to-go in dynamic environments with mixed-agents. In this chapter, we propose to learn a policy directly informing which actions lead to higher rewards.

RL Formulation

As in [111], the observation vector is composed by the ego-agent and the surrounding agents states, defined as:

$$\begin{aligned} \mathbf{s}_t &= \left[d_{\mathrm{g}}, \mathbf{p}_t - \mathbf{g}, v_{\mathrm{ref}}, \psi, r\right] \quad \text{(Ego-agent)} \\ \mathbf{s}_t^i &= \left[\mathbf{p}_t^i, \mathbf{v}_t^i, r^i, d_t^i, r^i + r\right] \quad \forall i \in \{1, n\} \quad \text{(Other agents)} \end{aligned} \tag{6.3}$$

where $\mathbf{s}_t$ is the ego-agent state and $\mathbf{s}_t^i$ the i-th agent state at step t. Moreover, $d_{\mathrm{g}} = \|\mathbf{p}_t - \mathbf{g}\|$ is the ego-agent's distance to goal and $d_t^i = \|\mathbf{p}_t - \mathbf{p}_t^i\|$ is the distance to the i-th agent.

Here, we seek to learn the optimal policy for the ego-agent $\pi : (\mathbf{s}_t, \mathrm{S}_t) \longrightarrow \mathbf{a}_t$ mapping the ego-agent's observation of the environment to a probability distribution of actions. We consider a continuous action space $\mathcal{A} \subset \mathbb{R}^2$ and define an action as position increments providing the direction maximizing the ego-agent rewards, defined as:

$$\mathbf{p}_t^{\mathrm{ref}} = \mathbf{p}_t + \delta_t \tag{6.4a}$$

$$\pi_{\theta^\pi}(\mathbf{s}_t, \mathrm{S}_t) = \delta_t = [\delta_{t,x}, \delta_{t,y}] \tag{6.4b}$$

$$\|\delta_t\| \leq N v_{\max}, \tag{6.4c}$$

where $\delta_{k,x}, \delta_{k,y}$ are the (x, y) position increments, $v_{\max}$ the maximum linear velocity and θ^π are the network policy parameters. Moreover, to ensure that the next sub-goal position is within the planning horizon of the ego-agent, we bound the action space according with

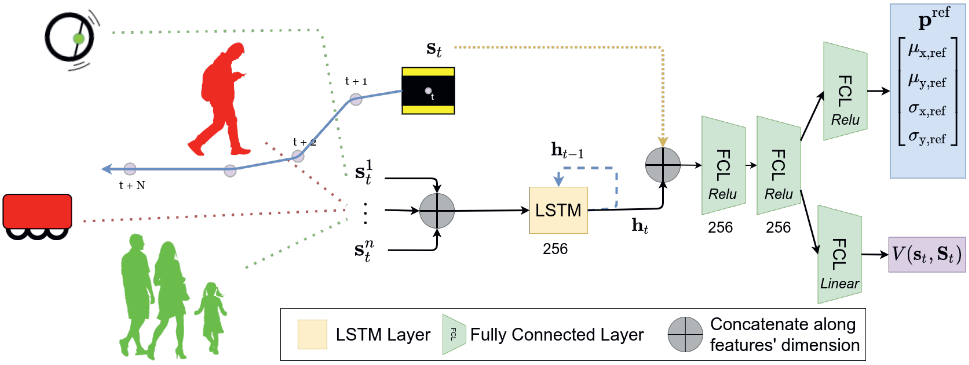


Figure 6.2: Proposed network policy architecture.

the planning horizon N of the optimization-based planner and its dynamic constraints, as represented in Equation (6.4b).

We design the reward function to motivate the ego-agent to reach the goal position while penalizing collisions:

$$R(\mathbf{s},\mathbf{a}) = \begin{cases} r_{\mathrm{goal}} & \text{if } \mathbf{p} = \mathbf{p}_g \\ r_{\mathrm{collision}} & \text{if } d_{\mathrm{min}} < r + r^i \ \forall i \in \{1, n\} \\ r_t & \text{otherwise} \end{cases} \tag{6.5}$$

where $d_{min} = \min_i \|\mathbf{p} - \mathbf{p}^i\|$ is the distance to the closest surrounding agent. r_t allows to adapt the reward function as shown in the ablation study (Sec.6.4.3), r_{goal} rewards the agent if reaches the goal $r_{\mathrm{collision}}$ penalizes if it collides with any other agents. In Section. 6.4.3 we analyze its influence in the behavior of the learned policy.

Policy Network Architecture

A key challenge in collision avoidance among pedestrians is that the number of nearby agents can vary between timesteps. Because feed-forward NNs require a fixed input vector size, prior work [17] proposed the use of Recurrent Neural Networks (RNNs) to compress the n agent states into a fixed size vector at each time-step. Yet, that approach discarded time-dependencies of successive observations (i.e., hidden states of recurrent cells).

Here, we use the “store-state” strategy, as proposed in [206]. During the rollout phase, at each time-step we store the hidden-state of the RNN together with the current state and other agents state, immediate reward and next state, $(\mathbf{s}_k, \mathbf{S}_k, \mathbf{h}_k, \mathbf{r}_k, \mathbf{s}_{k+1})$. Moreover, the previous hidden-state is feed back to warm-start the RNN in the next step, as depicted in Fig.6.2. During the training phase, we use the stored hidden-states to initialize the network. Our policy architecture is depicted in Figure 6.2. We employ a RNN to encode a variable sequence of the other agents states $\mathbf{S}_k$ and model the existing time-dependencies. Then, we concatenate the fixed-length representation of the other agent's states with the ego-agent's state to create a join state representation. This representation vector is fed to two fully-connected layers (FCL). The network has two output heads: one estimates the probability distribution parameters $\pi_{\theta^{\pi}}(\mathbf{s}, \mathbf{S}) \sim \mathcal{N}(\mu, \sigma)$ of the policy's action space and the other estimates the state-value function $V^{\pi}(s_t) := \mathbb{E}_{s_{t+1:\infty}}, \left[\sum_{l=0}^{\infty} r_{t+l}\right]$. μ and σ are the mean and variance of the policy's distribution, respectively.

6.3.2 Local Collision Avoidance

Here, we employ MPC to generate locally optimal commands respecting the kino-dynamics and collision avoidance constraints. To simplify the notation used, hereafter, we assume the current time-step t as zero.

State and Control Inputs

We define the ego-agent control input vector as $\mathbf{u} = [u_a, u_\alpha]$ and the control state as $\mathbf{x} = [x, y, \psi, v, w] \in \mathbb{R}^5$ following the dynamics model defined in Section 6.2.2.

Dynamic Collision Avoidance

We define a set of nonlinear constraints to ensure that the MPC generates collision-free control commands for the ego-agent (if a feasible solution exists). To limit the problem complexity and ensure to find a solution in real-time, we consider a limited number of surrounding agents $\mathcal{X}^m$, with $m \leq n$. Consider $\mathcal{X}^n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ as the set of all surrounding agent states, than the set of the m-th closest agents is:

***Definition* 6.1.** *A set $\mathcal{X}^m \subseteq \mathcal{X}^n$ is the set of the m-th closest agents if the euclidean distance* $\forall \mathbf{x}_j \in \mathcal{X}^m, \forall \mathbf{x}_i \in \mathcal{X}^n \setminus \mathcal{X}^m : \left\|\mathbf{x}_j, \mathbf{x}\right\| \leq \left\|\mathbf{x}_i, \mathbf{x}\right\|$.

We represent the area occupied by each agent $\mathcal{O}^i$ as a circle with radius r_i. To ensure collision-free motions, we impose that each circle $i \in \{1, \ldots, n\}$ i does not intersect with the area occupied by the ego-agent resulting in the following set of inequality constraints:

$$c_k^i(\mathbf{x}_k, \mathbf{x}_k^i) = \left\|\mathbf{p}_k, \mathbf{p}_k^i\right\| \geq r + r_i, \tag{6.6}$$

for each planning step k. This formulation can be extended for agents with general quadratic shapes, as in [202].

Cost Function

The subgoal recommender provides a reference position $\mathbf{p}_0^{\text{ref}}$ guiding the ego-agent toward the final goal position $\mathbf{g}$ and minimizing the cost-to-go while accounting for the other agents. The terminal cost is defined as the normalized distance between the ego-agent's terminal position (after a planning horizon N) and the reference position (with weight coefficient Q_N):

$$J_N(\mathbf{p}_N, \pi(\mathbf{x}, \mathbf{X})) = \left\| \frac{\mathbf{p}_N - \mathbf{p}_0^{\text{ref}}}{\mathbf{p}_0 - \mathbf{p}_0^{\text{ref}}} \right\|_{Q_N}, \tag{6.7}$$

To ensure smooth trajectories, we define the stage cost as a quadratic penalty on the ego-agent control commands

$$J_k^{\mathbf{u}}(\mathbf{u}_k) = \|\mathbf{u}_k\|_{Q_u}, \quad k = \{0, 1, \ldots, N-1\}, \tag{6.8}$$

where Q_u is the weight coefficient.

MPC Formulation

The MPC is then defined as a non-convex optimization problem

$$\begin{aligned} \min_{\mathbf{x}_{1:N}, \mathbf{u}_{0:N-1}} \quad & J_N(\mathbf{x}_N, \mathbf{p}_0^{\text{ref}}) + \sum_{k=0}^{N-1} J_k^u(\mathbf{u}_k) \\ \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{x}(0), \quad (6.1\text{d}), (6.2), \\ & c_k^i(\mathbf{x}_k, \mathbf{x}_k^i) > r + r_i, \\ & \mathbf{u}_k \in \mathcal{U}, \quad \mathbf{x}_k \in \mathcal{S}, \\ & \forall i \in \{1, \ldots, n\}; \forall k \in \{0, \ldots, N-1\}. \end{aligned} \tag{6.9}$$

Here, we assume a constant velocity model estimate of the other agents' future positions, as in [202].

6.3.3 PPO-MPC

We train the subgoal policy using a state-of-art method, Proximal Policy Optimization (PPO) [41], but the overall framework is agnostic to the specific RL training algorithm. In addition, we propose to jointly train the guidance policy π_{θ^π} and value function $V_{\theta^V}(\mathbf{s})$ *with* the MPC, as opposed to prior works [17] that use an idealized low-level controller during policy training (that cannot be implemented on a real robot). Algorithm 5 describes the proposed training strategy and has two main phases: supervised and RL training. First, we randomly initialize the policy and value function parameters $\{\theta^\pi, \theta^V\}$. Then, at the beginning of each episode we randomly select the number of surrounding agents between $[1, n_{\text{agents}}]$, the training scenario and the surrounding agents policy. More details about the different training scenarios and n_{agents} considered is given in Sec.6.4.2.

An initial RL policy is unlikely to lead an agent to a goal position. Hence, during the warm-start phase, we use the MPC as an expert and perform supervised training to train the policy and value function parameters for n_{MPC} steps. By setting the MPC goal state as the ego-agent final goal state $\mathbf{p}^{\text{ref}} = \mathbf{g}$ and solving the MPC problem, we obtain a locally optimal sequence of control states $\mathbf{x}_{1:N}^*$. For each step, we define $\mathbf{a}_t^* = \mathbf{x}_{t,N}^*$ and store the tuple containing the network hidden-state, state, next state, and reward in a buffer $\mathcal{B} \leftarrow \{\mathbf{s}_k, \mathbf{a}_t^*, r_k, \mathbf{h}_k, \mathbf{s}_{k+1}\}$. Then, we compute advantage estimates [40] and perform a supervised training step

$$\theta_{k+1}^V = \arg\min_{\theta^V} \mathbb{E}_{(\mathbf{a}_k, \mathbf{s}_k, r_k) \sim \mathcal{D}_{\text{MPC}}} \left[\left\| V_\theta(\mathbf{s}_k) - V_k^{\text{targ}} \right\| \right] \tag{6.10}$$

$$\theta_{k+1}^\pi = \arg\min_{\theta} \mathbb{E}_{(\mathbf{a}_k^*, \mathbf{s}_k) \sim \mathcal{D}_{\text{MPC}}} \left[\|\mathbf{a}_k^* - \pi_\theta(\mathbf{s}_k)\| \right] \tag{6.11}$$

Algorithm 5 PPO-MPC Training

```
**Inputs:** planning horizon $H$, value function and policy parameters $\{\theta^V, \theta^\pi\}$, number of supervised and RL training episodes $\{n_{\text{MPC}}, n_{\text{episodes}}\}$, number of agents $n$, $n_{\text{mini-batch}}$, and reward function $R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{a}_{t+1})$
Initialize states: $\{\mathbf{s}_0^0, \ldots, \mathbf{s}_0^n\} \sim \mathcal{S}$ , $\{\mathbf{g}^0, \ldots, \mathbf{g}^n\} \sim \mathcal{S}$
**while** $episode < n_{\text{episodes}}$ **do**
    Initialize $\mathcal{B} \longleftarrow \emptyset$ and $h_0 \longleftarrow \emptyset$
    **for** $k = 0, \ldots, n_{\text{mini-batch}}$ **do**
        **if** $episode \leq n_{\text{MPC}}$ **then**
            Solve Eq.6.9 considering $\mathbf{p}^{\text{ref}} = \mathbf{g}$
            Set $\mathbf{a}_t^* = \mathbf{x}_N^*$
        **else**
            $\mathbf{p}^{\text{ref}} = \pi_\theta(\mathbf{s}_t, \mathrm{S}_t)$
        **end if**
        $\{\mathbf{s}_k, \mathbf{a}_k, r_k, \mathbf{h}_{k+1}, \mathbf{s}_{k+1}, \text{done}\} = \text{Step}(\mathbf{s}_t^*, \mathbf{a}_t^*, \mathbf{h}_t)$
        Store $\mathcal{B} \longleftarrow \{\mathbf{s}_k, \mathbf{a}_k, r_k, \mathbf{h}_{k+1}, \mathbf{s}_{k+1}, \text{done}\}$
        **if** done **then**
            $episode += 1$
            Reset hidden-state: $h_t \longleftarrow \emptyset$
            Initialize: $\{\mathbf{s}_0^0, \ldots, \mathbf{s}_0^n\} \sim \mathcal{S}, \{\mathbf{g}^0, \ldots, \mathbf{g}^n\} \sim \mathcal{S}$
        **end if**
    **end for**
    **if** $episode \leq n_{\text{MPC}}$ **then**
        Supervised training: Eq.6.10 and Eq.6.11
    **else**
        PPO training [41]
    **end if**
**end while**
**return** $\{\theta^V, \theta^\pi\}$
```

where θ^V, θ^π are the value function and policy parameters, respectively. Note that θ^V and θ^π share the same parameter except for the final layer, as depicted in Fig.6.2. Afterwards, we use Proximal Policy Optimization (PPO) [41] with clipped gradients for training the policy. PPO is a on-policy method addressing the high-variance issue of policy gradient methods for continuous control problems. We refer the reader to [41] for more details about the method's equations. Please note that our approach is agnostic to which RL algorithm we use. Moreover, to increase the learning speed during training, we gradually increase the number of agents in the training environments (curriculum learning [207]).

6.4 Results

This section quantifies the performance throughout the training procedure, provides an ablation study, and compares the proposed method (sample trajectories and numerically) against the following baseline approaches:

- MPC: Model Predictive Controller from Section 6.3.2 with final goal position as position reference, $\mathbf{p}_{\text{ref}} = \mathbf{g}$;
- DRL [17]: state-of-the-art Deep Reinforcement Learning approach for multi-agent collision avoidance.

To analyze the impact of a realistic kinematic model during training, we consider two variants of the DRL method [17]: the same RL algorithm [17] was used to train a policy under a first-order unicycle model, referred to as DRL, and a second-order unicycle model (Eq.6.2), referred to as DRL-2. All experiments use a second-order unicycle model (Eq.6.2) in environments with cooperative and non-cooperative agents to represent realistic robot/pedestrian behavior.

6.4.1 Experimental Setup

The proposed training algorithm builds upon the open-source PPO implementation provided in the Stable-Baselines [208] package. We used a laptop with an Intel Core i7 and 32 GB of RAM for training. To solve the non-linear and non-convex MPC problem of Equation (6.9), we used the ForcesPro [168] solver. If no feasible solution is found within the maximum number of iterations, then the robot decelerates. All MPC methods used in this work consider collision constraints with up to the closest six agents so that the optimization problem can be solved in less than 20ms. Moreover, our policy's network has an average computation time of 2ms with a variance of 0.4ms for all experiments. Hyperparameter values are summarized in Table 6.1.

Table 6.1: Hyper-parameters.

Planning Horizon N	2 s	Number of mini batches	2048
Number of Stages	20	r_{goal}	3
γ	0.99	$r_{\text{collision}}$	-10
Clip factor	0.1	Learning rate	10^{-4}

6.4.2 Training Procedure

To train and evaluate our method we have selected four navigation scenarios, similar to [17, 18, 113]:

- **Symmetric swapping**: Each agent's position is randomly initialized in different quadrants of the $\mathbb{R}^2$ x-y plane, where all agents have the same distance to the origin. Each agent's goal is to swap positions with an agent from the opposite quadrant.
- **Asymmetric swapping**: As before, but all agents are located at different distances to the origin.
- **Pair-wise swapping**: Random initial positions; pairs of agents' goals are each other's intial positions
- **Random**: Random initial & goal positions

Each training episode consists of a random number of agents and a random scenario. At the start of each episode, each other agent's policy is sampled from a binomial distribution (80% cooperative, 20% non-cooperative). Moreover, for the cooperative agents we randomly sample a cooperation coefficient $c^i \sim \mathcal{U}(0.1, 1)$ and for the non-cooperative agents is randomly assigned a CV or non-CV policy (i.e., sinusoid or circular). Figure 6.3 shows the evolution of the robot average reward and the percentage of failure episodes. The top sub-plot compares our method average reward with the two baseline methods: DRL (with pre-trained weights) and MPC. The average reward for the baseline methods (orange, yellow) drops as the number of agents increases (each vertical bar). In contrast, our method (blue) improves with training and eventually achieves higher average reward for 10-agent scenarios than baseline methods achieve for 2-agent scenarios. The bottom plot demonstrates that the percentage of collisions decreases throughout training despite the number of agents increasing.

6.4.3 Ablation Study

A key design choice in RL is the reward function; here, we study the impact on policy performance of three variants of reward. The *sparse* reward uses $r_t = 0$ (only non-zero reward upon reaching goal/colliding). The *time* reward uses $r_t = -0.01$ (penalize every step until reaching goal). The *progress* reward uses $r_t = 0.01 * (\|\mathbf{s}_t - \mathbf{g}\| - \|\mathbf{s}_{t+1} - \mathbf{g}\|)$ (encourage motion toward goal). Aggregated results in Table 6.2 show that the resulting policy trained with a time reward function allows the robot to reach the goal with minimum time, to travel the smallest distance, and achieve the lowest percentage of failure cases. Based on these

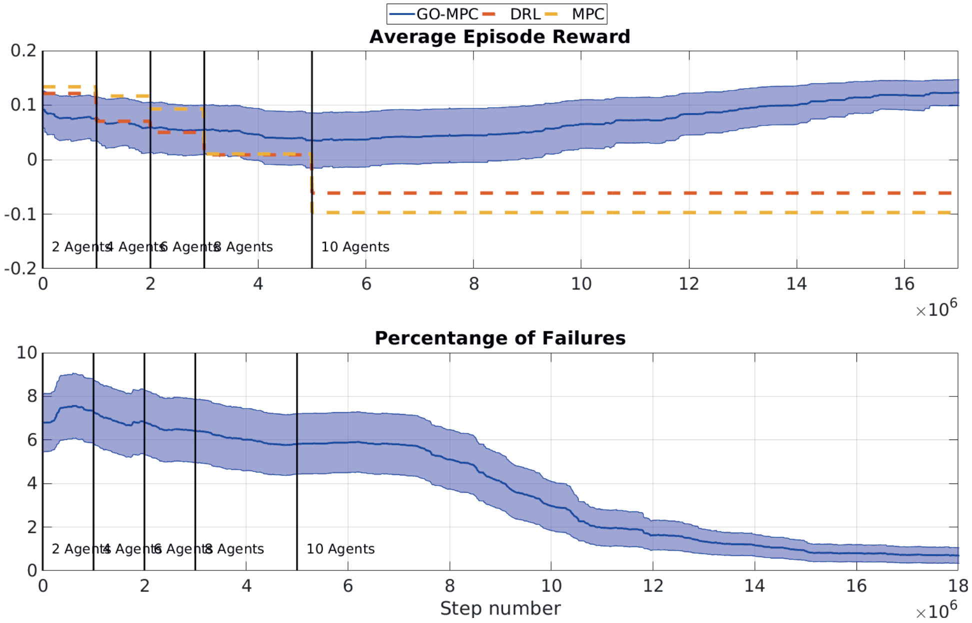


Figure 6.3: Moving average rewards and percentage of failure episodes during training. The top plot shows our method average episode reward vs DRL [17] and simple MPC.

results, we selected the policy trained with the time reward function for the subsequent experiments.

Table 6.2: Ablation Study: Discrete reward function leads to better policy than sparse, dense reward functions. Results are aggregated over 200 random scenarios with $n \in \{6, 8, 10\}$ agents.

	Time to Goal [s]			% failures (% collisions / % timeout)			Traveled distance Mean [m]		
# agents	6	8	10	6	8	10	6	8	10
Sparse Reward	8.00	8.51	8.52	0 (0 / 0)	1 (0 / 1)	2 (1 / 1)	13.90	14.34	14.31
Progress Reward	8.9	8.79	9.01	2 (1 / 1)	3 (3 / 0)	1 (1 / 0)	14.75	14.57	14.63
Time Reward	**7.69**	**8.03**	**8.12**	**0 (0 / 0)**	**0 (0 / 0)**	**0 (0 / 0)**	**13.25**	**14.01**	**14.06**

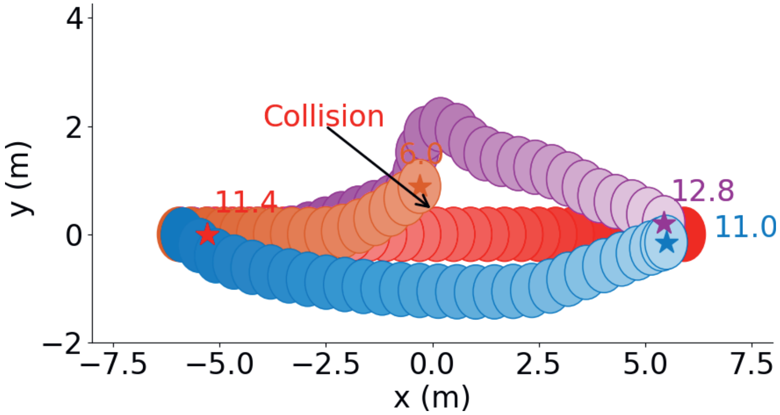
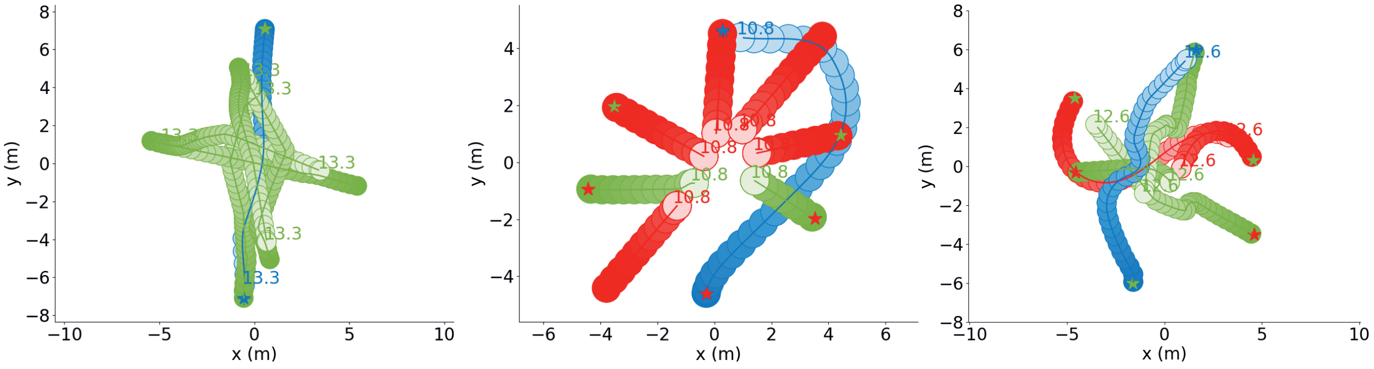


Figure 6.4: Two agents swapping scenario. In blue is depicted the trajectory of robot, in red the non-cooperative agent, in purple the DRL agent and, in orange the MPC.

6.4.4 Qualitative Analysis

This section compares and analyzes trajectories for different scenarios. Figure 6.4 shows that our method resolves a failure mode of both RL and MPC baselines. The robot has to swap position with a non-cooperative agent (red, moving right-to-left) and avoid a collision. We overlap the trajectories (moving left-to-right) performed by the robot following our method (blue) versus the baseline policies (orange, magenta). The MPC policy (orange) causes a collision due to the dynamic constraints and limited planning horizon. The DRL policy avoids the non-cooperative agent, but due to its reactive nature, only avoids the non-cooperative agent when very close, resulting in larger travel time. Finally, when using our approach, the robot initiates a collision avoidance maneuver early enough to lead to a smooth trajectory and faster arrival at the goal.

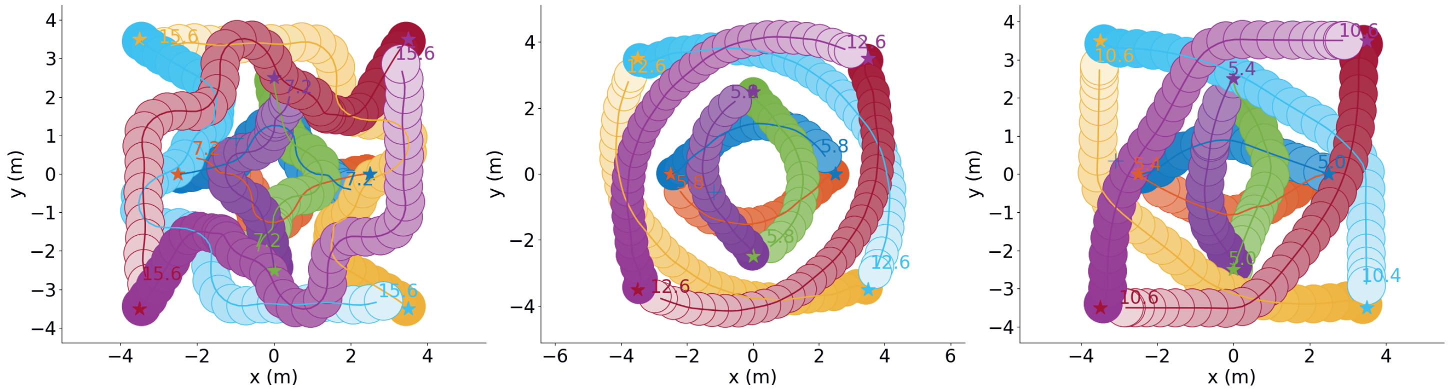


(a)

(b)

(c)

Figure 6.5: Sample trajectories with mixed agent policies (robot: blue, cooperative: green, non-cooperative: red). In (a), all agents are cooperative; in (b), two are cooperative and five non-cooperative (const. vel.); in (c), three are cooperative and two non-cooperative (sinusoidal). The GO-MPC agent avoids non-cooperative agents differently than cooperative agents.



(a) DRL [17]

(b) DRL-2 (ext. of [17])

(c) GO-MPC

Figure 6.6: *8 agents swapping positions. To simulate a multi-robot environment, all agents follow the same policy.*

We present results for mixed settings in Figure 6.5 and homogeneous settings in Figure 6.6 with $n \in \{6, 8, 10\}$ agents. In mixed settings, the robot follows our proposed policy while the other agents either follow an RVO [205] or a non-cooperative policy (same distribution as in training). Figure 6.5 demonstrates that our navigation policy behaves differently when dealing with only cooperative agents or both cooperative and non-cooperative. Whereas in Figure 6.5a the robot navigates through the crowd, Figure 6.5b shows that the robot takes a longer path to avoid the congestion.

In the homogeneous setting, all agents follow our proposed policy. Figure 6.6 shows that our method achieves faster time-to-goal than two DRL baselines. Note that this scenario was never introduced during the training phase, nor have the agents ever experienced other agents with the same policy before. Following the DRL policy (Figure 6.6a), all agents navigate straight to their goal positions leading to congestion in the center with reactive avoidance. The trajectories from the DRL-2 approach (Figure 6.6b) are more conservative, due to the limited acceleration available. In contrast, the trajectories generated by our approach (Figure 6.6c), present a balance between going straight to the goal and avoiding congestion in the center, allowing the agents to reach their goals faster and with smaller distance traveled.

6.4.5 Performance Results

This section aggregates performance of the various methods across 200 random scenarios. Performance is quantified by average time to reach the goal position, percentage of episodes that end in failures (either collision or timeout), and the average distance traveled.

The numerical results are summarized in Table 6.3. Our method outperforms each baseline for both mixed and homogeneous scenarios. To evaluate the statistical significance, we performed pairwise Mann–Whitney U-tests between GO-MPC and each baseline (95% confidence). GO-MPC shows statistically significant performance improvements over the DRL-2 baseline in terms of travel time and distance, and the DRL baseline in term of travel time for six agents and travel distance for ten agents. For homogeneous scenarios, GO-MPC is more conservative than DRL and MPC baselines resulting in a larger average traveled distance. Nevertheless, GO-MPC is reaches the goals faster than each baseline and is less conservative than DRL-2, as measured by a significantly lower average distance traveled.

Finally, considering higher-order dynamics when training DRL agents (DRL-2) improves the collision avoidance performance. However, it also increases the average time to goal and traveled distance, meaning a more conservative policy that still under-performs GO-MPC in each metric.

Table 6.3: Statistics for 200 runs of proposed method (GO-MPC) compared to baselines (MPC, DRL [17] and DRL-2, an extension of [17]): time to goal and traveled distance for the successful episodes, and number of episodes resulting in collision for $n \in \{6, 8, 10\}$ agents. For the mixed setting, 80% of agents are cooperative, and 20% are non-cooperative.

	Time to Goal (mean ± std) [s]			% failures (% collisions / % deadlocks)			Traveled Distance (mean ± std) [m]		
# agents	6	8	10	6	8	10	6	8	10
Mixed Agents									
MPC	11.2 ± 2.2	11.3 ± 2.4	11.0 ± 2.2	13 (0 / 0)	22 (0 / 0)	22 (22 / 0)	12.2 ± 2.3	12.4 ± 2.5	12.1 ± 2.3
DRL [17]	13.7 ± 3.0	13.7 ± 3.1	14.4 ± 3.3	17 (17 / 0)	23 (23 / 0)	29 (29 / 0)	13.8 ± 3.3	13.8 ± 4.0	14.4 ± 3.3
DRL-2 [17]+	15.3 ± 2.3	16.1 ± 2.2	16.7 ± 2.2	6 (6 / 0)	10 (10 / 0)	13 (13 / 0)	14.9 ± 2.3	16.1 ± 2.2	16.7 ± 2.2
GO-MPC	12.7 ± 2.7	12.9 ± 2.8	13.3 ± 2.8	**0 (0 / 0)**	**0 (0 / 0)**	**0 (0 / 0)**	13.7 ± 2.7	13.8 ± 2.8	14.3 ± 2.8
Homogeneous									
MPC	17.37 ± 2.9	16.38 ± 1.5	16.64 ± 1.7	30 (29 / 1)	36 (25 / 11)	35 (28 / 7)	11.3 ± 2.1	10.9 ± 2.3	10.6 ± 2.8
DRL [17]	14.2 ± 2.4	14.4 ± 2.7	14.6 ± 3.3	16 (14 / 2)	20 (18 / 2)	20 (20 / 0)	12.8 ± 2.3	12.2 ± 2.3	12.2 ± 3.2
DRL-2 [17]+	15.9 ± 3.1	17.5 ± 4.2	15.9 ± 4.5	17 (11 / 6)	29 (21 / 8)	28 (24 / 4)	15.2 ± 3.0	15.9 ± 4.2	15.4 ± 4.5
GO-MPC	**13.8 ± 2.9**	**14.3 ± 3.3**	**14.6 ± 2.9**	**0 (0 / 0)**	**0 (0 / 0)**	**2 (1 / 1)**	14.7 ± 2.9	15.1 ± 3.3	15.1 ± 2.9

6.5 Conclusions and Future Work

This chapter introduced a subgoal planning policy for guiding a local optimization planner. We employed DRL methods to learn a subgoal policy accounting for the interaction effects among the agents. Then, we used an MPC to compute locally optimal motion plans respecting the robot dynamics and collision avoidance constraints. Learning a subgoal policy improved the collision avoidance performance among cooperative and non-cooperative agents as well as in multi-robot environments. Moreover, our approach can reduce travel time and distance in cluttered environments. Future work could account for environment constraints.

7

Learning Interaction-Aware Guidance for Trajectory Optimization in Dense Traffic Scenarios

---

This chapter is based on:

- B. Brito, A. Agarwal, and J. Alonso-Mora, "Learning Interaction-Aware Guidance for Trajectory Optimization in Dense Traffic Scenarios," in *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, doi: 10.1109/TITS.2022.3160936.

Code: https://github.com/tud-amr/int-mpc

7.1 Introduction

In the previous chapter (Chapter 6), we have demonstrated that deep reinforcement learning algorithms can be employed to learn a policy providing global guidance (i.e., next subgoal) to a local optimization-based planner through its cost function. However, for autonomous navigation in dense traffic scenarios providing the next optimal subgoal position is not sufficient. In dense traffic scenarios, autonomous vehicles (AVs) must engage in a non-verbal negotiation and exploit the interactions with each other t efficiently and safely navigate. Taking inspiration in Chapter 6, in this chapter we introduce a learning-based method enabling interactive behavior for navigation in dense traffic scenarios.

Despite recent advancements in autonomous driving solutions (e.g., Waymo [209], Uber [210]), driving in real-world dense traffic scenarios such as highway merging and unprotected left turns still stands as a hurdle in the widespread deployment of autonomous vehicles [8]. Driving in dense traffic conditions is intrinsically an interactive task [211], where the AVs' actions elicit immediate reactions from nearby traffic participants and vice-versa. An example of such behavior is illustrated in Fig. 7.1, where the autonomous vehicle needs to perform a merging maneuver onto the main lane. To accomplish this task, it needs to first reason about the other driver's intentions (e.g., to yield or not to yield) without any explicit inter-vehicle communication. Then, it needs to know how to interact with multiple road-users and leverage other vehicles' cooperativeness to induce them to yield, such that they create room for the AV to merge safely.

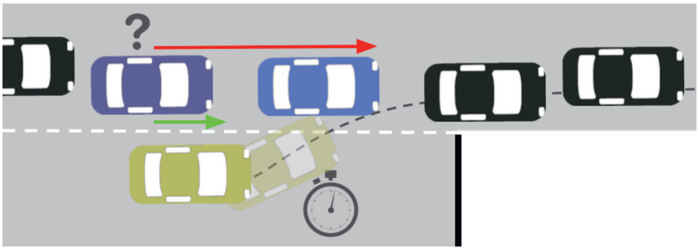


Figure 7.1: Illustration of a dense on-ramp merging traffic scenario where the autonomous vehicle (yellow) needs to interact with other traffic participants in order to merge onto the main lane in a timely and safe manner. The potential follower (purple) may yield (green arrow) to the autonomous vehicle leaving space for the autonomous vehicle to merge or behave non-cooperatively (red arrow) to deter the autonomous vehicle from merging. To successfully merge, the autonomous vehicle needs to identify the cooperative ones by interacting with them without any explicit inter-vehicle communication.

The development of interaction-aware prediction models has been studied [170, 212], allowing AVs to reason about other drivers' intentions. In contrast, developing interactive motion planning algorithms that can reason and exploit other drivers cooperativeness is still challenging [14]. The majority of traditional motion planning methods are too conservative and fail in dense scenarios because they do not account for the interaction between the autonomous vehicle and nearby traffic [8], [213]. However, works that account for the interaction among agents do not scale for many agents due to the curse of dimensionality

[16, 155, 214]. Deep Reinforcement Learning (DRL) methods can overcome the latter, but either do not provide any safety guarantees [162] or are overly conservative to ensure safety [166].

In this chapter, we introduce an interactive Model Predictive Controller (IntMPC) for safe navigation in dense traffic scenarios. We explore the insight that human drivers communicate their intentions and negotiate their driving maneuvers by adjusting both distance and time headway to the other vehicles [215, 216]. Studies show that in dense traffic scenarios, such as merging and left-turning, cooperative or aggressive behavior is strongly connected to higher or smaller average distance and time headway [217, 218], respectively. These driving features (i.e., relative distance and time headway) can be directly translated into a velocity reference. Hence, we propose to learn, via Deep Reinforcement Learning (DRL), an interaction-aware policy as a velocity reference. This reference provides global guidance to a local optimization-based planner, which ensures that the generated trajectories are kino-dynamically feasible and safety constraints are respected. Our method leverages vehicles' interaction effects to create free-space areas for the AV to navigate and complete various driving maneuvers in cluttered environments. The main contribution of this work is an Interactive Model Predictive Controller (IntMPC) for navigation in dense traffic environments combining DRL to learn an interaction-aware policy providing global guidance (velocity reference) in the cost function to a local optimization-based planner.

Extensive simulation results demonstrate that our approach triggers interactive negotiating behavior to reason about the other drivers' cooperation and exploit their cooperativeness to induce them to yield while remaining safe.

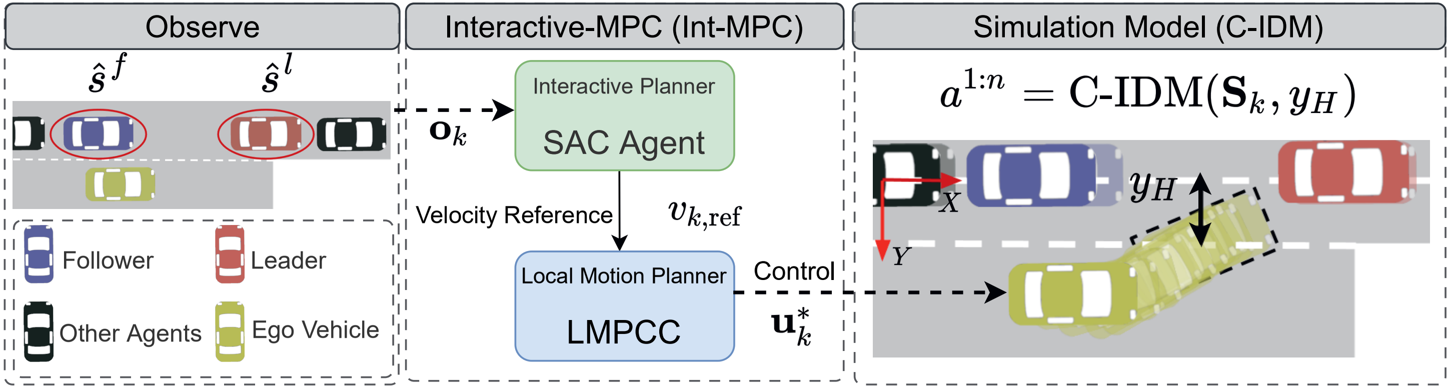


Figure 7.2: Our proposed architecture comprises of three main modules: an Interactive Reinforcement Learner (DRL Agent), a Local Motion Planner (MPCC) and a Simulation Model (P-IDM). The AV observes the leader state s^l and follower state s^f relative to it, which serves as input to the Interactive Planner providing a reference velocity $v_{k,ref} = \pi(\mathbf{s}_k, \mathbf{S}_k)$ for the MPCC to follow. The MPCC then computes locally optimal sequence of control commands $\mathbf{u}^*_{0:H-1}$ minimizing a cost function $J(\mathbf{s}_k, \mathbf{u}_k)$ (See Section 7.3.2). The reference velocity v_{ref} allows to directly control the AV aggressiveness and thus, to control the interaction with the other vehicles. Finally, P-IDM then computes acceleration command for the other vehicles based on the estimated AV's motion plan (Section 7.5.8).

7.2 Problem Formulation

Consider a set $\mathcal{X}$ of n vehicles interacting in a dense traffic scenario comprising an autonomous vehicle (AV) and $n-1$ human drivers, henceforth referred to as other vehicles, exhibiting different levels of willingness to yield. The term "vehicles" is used to collectively refer to the AV and other vehicles. At the beginning of an episode, the AV receives a global reference path $\mathcal{P}$ to follow from a path planner consisting of a sequence of M waypoints $\mathbf{p}^r_m = [x^r_m, y^r_m] \in \mathbb{R}^2$ with $m \in \mathcal{M} := \{1,\dots,M\}$. For each time-step k, the AV observes its state $\mathbf{s}_k$ and the states of other agents $\mathrm{S}_k = [\mathbf{s}^1_k,\dots,\mathbf{s}^{n-1}_k]$, then takes action $\mathbf{a}_k$, leading to the immediate reward $R(\mathbf{s}_k,\mathbf{a}_k)$ and next state $\mathbf{s}_{k+1} = f(\mathbf{s}_k,\mathbf{u}_k)$, under the dynamic model f[1] and controller model h, with $\mathbf{u}_k = h(\mathbf{s}_k,\mathbf{a}_k)$. The vehicle's state is defined as

$$\mathbf{s}^i_k = \{x_k, y_k, \psi_k, v_k\} \forall i \in \{0,\dots,n-1\}$$

where x_k and y_k are the Cartesian position coordinates, ψ_k the heading angle and v_k the forward velocity in a global inertial frame $\mathcal{W}$ fixed in the main lane (see Figure 7.2). $\mathcal{A}^{\text{ego}}$ and $\mathcal{A}^i$ denote the area occupied by the AV and the i-th other vehicle, respectively. We aim to learn a continuous policy $\pi(\mathbf{a}_k|\mathbf{s}_k,\mathrm{S}_k)$ conditioned on the AV's and other vehicles' states minimizing the expected driving time $\mathbb{E}[t_g]$ for the AV to reach its goal position while ensuring collision-free motions, defined as the following optimization problem:

$$\begin{aligned}
\pi^* = &\underset{\pi}{\operatorname{argmin}} \quad \mathbb{E}\left[t_g \mid \pi(\mathbf{a}_k|\mathbf{s}_k,\mathrm{S}_k)\right] \\
\text{s.t.} \quad & \mathbf{s}_{k+1} = f(\mathbf{s}_k,\mathbf{u}_k), && \text{(7.1a)} \\
& \mathbf{u}_k = h(\mathbf{s}_k,\pi(\mathbf{a}_k|\mathbf{s}_k,\mathrm{S}_k)) && \text{(7.1b)} \\
& \mathcal{A}^{\text{ego}}_k \cap \mathcal{A}^i_k = \emptyset && \text{(7.1c)} \\
& \mathbf{u}_k \in \mathcal{U},\ \mathbf{s}_k \in \mathcal{S},\ \mathbf{a}_t \in \mathcal{A}, && \text{(7.1d)} \\
& \forall i \in \{1\dots n-1\}\ \forall k \in \{0\dots t_g\}
\end{aligned}$$

where (7.1a) are the kino-dynamic constraints, (7.1c) the collision avoidance constraints, and $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{U}$ are the set of admissible states, actions, and control inputs (e.g., maximum vehicles' speed), respectively. We assume that each vehicle's current position and velocity are observed (e.g., from on-board sensor data) and no inter-vehicle communication.

---

[1] This is identical to the Vehicle Model used in the simulation defined in Section 7.3.2

7.3 Interactive Model Predictive Control

This section introduces the proposed Interactive Model Predictive Control (IntMPC) framework for safe navigation in dense traffic scenarios. Figure 7.2 depicts our proposed motion planning architecture incorporating three main modules: an interactive reinforcement learner, a local optimization planner, and an interactive simulation environment. Firstly, we define the RL framework to learn an interaction-aware navigation policy (Section 7.3.1), providing global guidance to a local optimization planner (Section 7.3.2). Secondly, we introduce our training algorithm to jointly train the interaction-aware policy and the local optimization planner (Section 7.3.3). Our IntMPC enhances the AV with interactive behavior, exploiting the other traffic participants' interaction effects.s To finalize ,we introduce the behavior module used to simulate dense traffic scenarios with various driving behavior, ranging from cooperative to non-cooperative. Here, we propose an expansion for the Intelligent Driver Model (IDM) model allowing the other vehicles to react to the other's predicted plans (Section 7.4).

7.3.1 Interactive Planner

Here, we propose to use deep RL to learn an interaction-aware velocity reference exploiting the interaction effects between the vehicles and providing global guidance to a local optimization-based planner.

RL Formulation

The AV's observation vector is composed by the leader's (vehicle in front) and the follower's (vehicle behind the AV) state, $\mathbf{o}_k = [\mathbf{s}_k^l, \mathbf{s}_k^f]$, relative to the AV's frame. To enable interactive behavior with the other traffic participants, we define the RL policy's action as a velocity reference to directly control the interaction at the merging point. High-speed values lead to more aggressive and low-speed to more conservative behavior, respectively. Hence, we consider a continuous action space $\mathcal{A} \subset \mathbb{R}$ and aim to learn the optimal policy π mapping the AV's state and observation to a probability distribution of actions.

$$\pi_\theta(\mathbf{s}_k, \mathbf{o}_k) = \mathbf{a}_k = v_{k,\mathrm{ref}} \tag{7.2a}$$

$$\pi_\theta(\mathbf{s}_k, \mathbf{o}_k) \sim \mathcal{N}(\mu_k, \sigma_k) \tag{7.2b}$$

where θ are the policy's network parameters, $\mathcal{N}$ is a multivariate Gaussian density function, and μ and σ are the Gaussian's mean and variance, respectively.

We formulate a reward function to motivate progress along a reference path, to penalize collisions and infeasible solutions, and when moving too close to another vehicle. The reward function is the summation of the four terms described as follows:

$$R(\mathbf{s}_k, \mathbf{o}_k, \mathbf{a}_k) = \begin{cases} v_k & \\ r_{\text{infeasible}} & \\ r_{\text{collision}} & \text{if } \mathcal{A}_k^{\text{ego}} \cap \mathcal{A}_k^{\text{i}} \neq \emptyset \\ r_{\text{near}} & d_{\min}(\mathbf{s}_k, \mathbf{s}_k^i) \leq \Delta d_{\min} \end{cases} \tag{7.3}$$

where $c_k^{c,i}$ is the collision avoidance constraint between the AV and the vehicle i (Section 7.3.2), $\mathcal{A}_k^{ego} \cap \mathcal{A}_k^i$ represents the common area occupied by the AV and the i-th other vehicle at step k. $d_{\min}$ is the minimum distance to the closest nearby vehicle i and $\Delta d_{\min}$ is a hyper-parameter distance threshold. The first term v_k is a reward proportional to the AV's velocity encouraging higher velocities and thus, encouraging interaction and minimizing the time to goal. The second $r_{\text{infeasible}}$, third $r_{\text{collision}}$ and fourth term r_{near} penalize the AV for infeasible solutions, collisions and for driving too close to other vehicles, respectively.

7.3.2 Local Motion Planner

Deep RL can be used to learn an end-to-end control policy in dense traffic scenarios [162], [158]. However, their sample inefficiency [219] and transferability issues [220] makes it hard to apply them in real-world settings. In contrast, optimization-based methods have been widely used and deployed into actual autonomous vehicles [143, 221]. Therefore, we employ Model Predictive Contour Control (MPCC) to generate locally optimal control commands following a reference path while satisfying kino-dynamics and collision avoidance constraints if a feasible solution is found. The reference path can be provided by a global path planner such as Rapidly-exploring Random Trees (RRT) [222].

Vehicle Model

We employ a kinematic bicycle model for the AV, described as follows:

$$\begin{aligned} \dot{x} &= v\cos(\phi + \beta) \\ \dot{y} &= v\sin(\phi + \beta) \\ \dot{\phi} &= \frac{v}{l_r}\sin(\beta) \\ \dot{v} &= u^a \\ \beta &= \arctan\left(\frac{l_r}{l_f + l_r}\tan\left(u^\delta\right)\right) \end{aligned} \tag{7.4}$$

where β is the velocity angle. The distances of the rear and front tires from the center of gravity of the vehicle are l_r and l_f, respectively, and are assumed to be identical for simplicity. The vehicle control input $\mathbf{u}$ is the forward acceleration u^a and steering angle u^δ, $\mathbf{u} = [u^a, u^\delta]$.

Cost Function

The local controller receives a velocity reference v_{ref}, from the Interactive Planner (Section 7.3.1), exploiting for the interaction effects of the AV in the other vehicles to maximize long-term rewards. To enable the AV to follow the reference path while tracking the velocity reference, we define the stage cost as follows:

$$J(\mathbf{s_k}, \mathbf{u}_k, \lambda_k) = \left\| e_k^c(\mathbf{s}_k, \lambda_k) \right\|_{q_c} + \left\| e_k^l(\mathbf{s}_k, \lambda_k) \right\|_{q_l} + \left\| v_{k,ref} - v_k \right\|_{q_v} + \left\| u_k^a \right\|_{q_u} + \left\| u_k^\delta \right\|_{q_\delta} \tag{7.5}$$

where $Q = \{q_c, q_l, q_v, q_u, q_\delta\}$ denotes the set of cost weights and λ_k is the estimated progress along the reference path. To track the reference path closely, we minimize two cost terms: the contour error (e_k^c) and lag error (e_k^l). Contour error gives a measure of how far the ego vehicle deviates from the reference path laterally whereas lag error measures the deviation of the ego vehicle from the reference path in the longitudinal direction. For more details on path parameterization and tracking error, please refer to [143]. The third term, $\|v_{k,\text{ref}} - v_k\|$, motivates the planner to follow v_{ref} closely. Finally, to generate smooth trajectories, we add a quadratic penalty to the control commands u_k^a and u_k^δ.

Dynamic Obstacle Avoidance

The occupied area by the AV, $\mathcal{A}^{\text{ego}}(s_k)$, is approximated with a union of n_c circles i.e $\bar{A}^{\text{ego}}(s_k) \subseteq \bigcup_{c \in \{1,\ldots,n_c\}} \mathcal{A}_c(s_k)$, where $\mathcal{A}_c$ is the area occupied for a circle with radius r. For each vehicle i, the occupied area $\mathcal{A}^i$ is approximated by an ellipse of semi-major axis a_i, semi-minor axis b_i and orientation ϕ. To ensure collision-free motions, we define a set of non-linear constraints imposing that each circle c of the AV with the elliptical area occupied by the i-th vehicle does not intersect:

$$c_k^{i,c}(\mathbf{s}_k, \mathbf{s}_k^i) = \begin{bmatrix} \Delta x_k^c \\ \Delta y_k^c \end{bmatrix}^{\mathrm{T}} R(\phi)^{\mathrm{T}} \begin{bmatrix} \frac{1}{\alpha^2} & 0 \\ 0 & \frac{1}{\beta^2} \end{bmatrix} R(\phi) \begin{bmatrix} \Delta x_k^c \\ \Delta y_k^c \end{bmatrix} > 1, \tag{7.6}$$

$\forall k \in \{0, \ldots, H\}$ and $\forall i \in \{1, \ldots, n-1\}$. The parameters Δx_k^c and Δy_k^c represent x-y relative distances in AV's frame between the disc c and the ellipse i for prediction step k. $R(\psi)$ is the rotation matrix. To guarantee collision avoidance we enlarge the other vehicle's semi-major and semi-minor axis with a r_{disc} coefficient, assuming $\alpha = a + r_{\text{disc}}$ and $\beta = b + r_{\text{disc}}$, as described in [223].

Road Boundaries

We introduce constraints on the lateral distance (i.e., contour error) of the AV with respect to the reference path to ensure that the vehicle stays within the road boundaries [142]:

$$-w_{\text{left}}^{\text{road}} \leq e_k^c(\mathbf{s}_k) \leq w_{\text{right}}^{\text{road}} \tag{7.7}$$

where $w_{\text{left}}^{\text{road}}$ and $w_{\text{right}}^{\text{road}}$ are the left and right road limits, respectively.

MPC Formulation

We formulate the motion planning problem as a Receding Horizon Trajectory Optimization problem (7.8) with planning horizon H conditioned on the following constraints:

$$
\begin{aligned}
\mathbf{u}^*_{0:H-1} = & \min_{u_{0:H-1}} \sum_{k=0}^{H-1} J(\mathbf{s}_k, \mathbf{u}_k, \lambda_k) + J(\mathbf{s}_H, \lambda_H) && \text{(7.8a)}\\
\text{s.t.} \quad & \mathbf{s}_{k+1} = f(\mathbf{s}_k, \mathbf{u}_k), && \text{(7.8b)}\\
& \lambda_{k+1} = \lambda_k + v_k \Delta t && \text{(7.8c)}\\
& -w^{\text{road}}_{\text{left}} \leq e^c(\mathbf{s}_k) \leq w^{\text{road}}_{\text{right}} && \text{(7.8d)}\\
& c^{i,c}_k(\mathbf{s}_k, \mathbf{s}^i_k) > 1 \quad \forall c \in \{1, \dots, n_c\}, && \text{(7.8e)}\\
& \mathbf{u}_k \in \mathcal{U}, \quad \mathbf{s}_k \in \mathcal{S}, && \text{(7.8f)}\\
& \forall k \in \{0, \dots, H\}. && \text{(7.8g)}
\end{aligned}
$$

where Δt is the discretization time and $\mathbf{u}^*_{0:H-1}$ the locally optimal control sequence for H time-steps. In this work, we assume a constant velocity model to estimate of the other vehicles' future positions, as in [223].

7.3.3 Training Procedure

We train the policy using Soft Actor-Critic (SAC) [46] to learn the policy's probability distribution parameters. SAC augments traditional RL algorithms' objective with the policy's entropy, embedding the notion of exploration into the policy while giving up on clearly unpromising paths [46]. We propose to jointly train the guidance policy with the local motion planner allowing the trained policy to directly implement our method on a real system and learn with the cases resulting in infeasible solutions for the optimization solver. In contrast to prior works on safe RL [165], during training, we do not employ collision constraints (Eq. 7.8e), exposing the policy to dangerous situations or collisions which is necessary to learn how to interact with other vehicles closely.

Algorithm 6 describes the proposed training strategy. Each episode begins with the initialization of all vehicle's states (see Sections 7.5.2 and 7.5.3 for more details). Every K cycles, we sample a reference velocity v_{ref} from the policy π_θ. Querying the interaction-aware policy every K control cycles helps to stabilize the training procedure and better assess the policy's impact on the environment (see Section 7.5.8). Then, the MPCC computes a locally optimal sequence of steering and acceleration commands $u^*_{0:H-1}$ for the AV. If a feasible solution is found, we apply the first control command of the sequence and re-compute the motion plan in the next cycle considering new observations. If no feasible solution is found, we apply a braking command. Training the interaction-aware policy with the MPCC controller enables the policy to account for the controller and AV constraints. Afterward, the P-IDM computes an action for each vehicle on the main lane while being aware of the AV on the adjacent lane. An episode is over if: the AV reaches the goal position (finishes merging or turning left); the AV collides with another vehicle; it does not finish the maneuver in time (i.e., timeout). Finally, to update the policy's distribution parameters, we employ the Soft Actor-Critic (SAC) [46] method. We refer the reader to [46] for more

Algorithm 6 Training Procedure

```
Inputs: planning horizon $H$, initial policy's parameters $\theta$, Q-functions' parameters $\{\phi_1, \phi_2\}$, number of training episodes $n_{\text{episodes}}$, number of vehicles $n$, reward function $R(\mathbf{s}_k, \mathbf{o}_k, \mathbf{a}_k)$ and number of control steps $K$
Initialize initial states: $\{\mathbf{s}_0, \ldots, \mathbf{s}_0^{n-1}\} \sim \mathcal{S}$
Initialize replay buffer: $\mathcal{D} \longleftarrow \emptyset$
while $episode < n_{\text{episodes}}$ do
    Get observation $\mathbf{o}_k$ and AV's state $\mathbf{s}_k$
    if $k$ mod $K == 0$ then
        Sample velocity reference for the AV:
        $v_{k,\text{ref}} \sim \pi_\theta(\mathbf{s}_k, \mathbf{o}_k)$
    end if
    Solve the optimization problem of Eq. 7.8 without collision constraints (Eq. 7.8e) :
    $\mathbf{u}^*_{k:k+H} = \text{MPCC}(v_{k,\text{ref}}, \mathbf{s}_k, \mathbf{o}_k)$
    Estimate AV's lateral position:
    $\tilde{y}_H = \text{PredictionModel}(v_k, \mathbf{s}_k, \mathbf{o}_k)$ (Section 7.5.8)
    $\{\mathbf{s}_{k+1}, \text{done}, \mathbf{r}_k\} = \text{Step}(\mathbf{s}_k, \mathbf{u}_k)$
    Store $(\mathbf{s}_k, \mathbf{a}_k, r_k, \mathbf{s}_{k+1}, \text{done})$ in replay buffer $\mathcal{D}$
    if done then
        $episode += 1$
        Initialize: $\{\mathbf{s}_0, \ldots, \mathbf{s}_0^{n}\} \sim \mathcal{S}$
    end if
    if it's time to update then
        SAC training [46]
    end if
end while
return $\{\theta, \phi_1, \phi_2\}$
```

details about the learning method's equations. Please note that our approach is agnostic to which RL algorithm we use.

7.3.4 Online Planning

Algorithm 7 describes our Interactive Model Predictive Controller (IntMPC) algorithm. For every step k, we first obtain a velocity reference, v_{ref}, from the trained policy. Then, by solving the MPCC problem (Equation (7.8)), we obtain a locally optimal sequence of control commands $\mathbf{u}^*_{k:k+H}$. Finally, if the MPCC plan is feasible we employ the first control command, $\mathbf{u}^*_k$, and re-compute a new plan considering the new observations following a receding horizon control strategy. Else, we apply a braking command, $\mathbf{u}_{\text{safe}}$.

Algorithm 7 Int-MPC

```
Inputs: AV's state $\mathbf{s}_k$, observation $\mathbf{o}_k$ and reference path $\mathbf{p}^r_m = [x^r_m, y^r_m] \in \mathbb{R}^2$ with
    $m \in \mathcal{M} := \{1, \ldots, M\}$ waypoints.
for $k = 0, 1, 2, \ldots$ do
    Get observation $\mathbf{o}_k$ and AV's state $\mathbf{s}_k$
    Sample velocity reference for the AV:
      $v_{k,\mathrm{ref}} = \pi_\theta(\mathbf{s}_k, \mathbf{o}_k)$
    Compute MPCC trajectory by solving Eq. 7.8:
      $\mathbf{u}^*_{k:k+H} = \mathrm{MPCC}(v_{k,\mathrm{ref}}, \mathbf{s}_k, \mathbf{o}_k)$
    if $\mathbf{u}^*_{k:k+H}$ is feasible then
        Apply $\mathbf{u}^*_k$
    else
        Apply $\mathbf{u}_{\mathrm{safe}}$
    end if
end for
```

7.4 Modeling Other Traffic Drivers' Behaviors

We aim to simulate dense and complex negotiating behavior with varying degrees of willingness to yield. For instance, in a typical dense traffic scenario (e.g., on-ramp merging), human drivers trying to merge onto the main lane need to leverage other drivers' cooperativeness to create obstacle-free space to merge safely. In contrast, drivers on the main lane exhibit different levels of willingness to yield. Some drivers stop as soon as they spot the other vehicle on the adjacent lane (Cooperative). Other drivers ignore the other vehicles entirely and may even accelerate to deter it from merging (Non-Cooperative). Moreover, they also consider an internal belief about the other vehicle's motion plan on the adjacent lane in their decision-making process at the merging point. Here, we introduce the Predictive Intelligent Driver Model (P-IDM) to control the longitudinal driving behavior of the other vehicles, built on the Intelligent Driver Model (IDM) [224]. Our proposed model consists of three main steps: leader and follower selection, other vehicles' motion estimation, and control command computation.

Leader & Follower Selection

For each vehicle, the model assigns a leader, denoted with up-script l, and a follower, denoted with up-script f. Consider $\mathcal{X}^l_i$ as the set of potential leaders for the vehicle i, then

***Definition* 7.1.** *IDM: A set* $\mathcal{X}^l_i \subseteq \mathcal{X}$ *is the set of possible leaders for the vehicle* i *if* $\forall j \in [0, n-1], j \neq i : x^j_k > x^i_k$ *and* $y^i_k < c_i$.

***Definition* 7.2.** *IDM: A set* $\mathcal{X}^f_i \subseteq \mathcal{X}$ *is the set of possible followers for the vehicle* i *if* $\forall j \in [0, n-1], j \neq i : x^j_k < x^i_k$ *and* $y^i_k < c_i$.

where c_i is a hyper-parameter threshold used to model cooperation (Section 7.5.3) [162]. Figure 7.3 shows an example of the leader's and follower's sets for the merging scenario

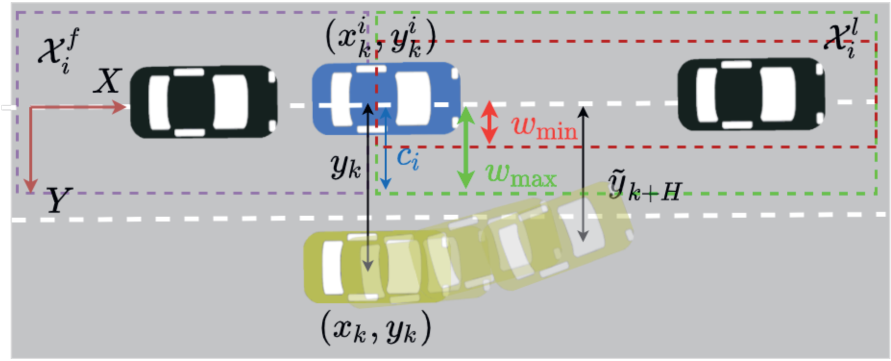


Figure 7.3: Leader & Follower Selection Process. The AV is depicted in yellow, the i-th interacting vehicle in blue, and the i-th vehicle's follower and leader in black. (x_k, y_k) *are the x-y position coordinates in the main lane frame of the AV and* (x_k^i, y_k^i) *of the i-th vehicle on the main lane at time-step k. Dashed purple represents the followers' area set. Dashed red and green represent the leader's area set. To model mixed driving behavior, the i-th vehicle cooperation coefficient* c_i *is randomly sampled from a uniform bounded distribution* $c_i \sim \mathcal{U}([w_{\min}, w_{\max}])$ *(defined in Section 7.5.3).* $w_{\max}$ *and* $w_{\min}$ *represents a maximum and minimum distance between the center of the current lane and the adjacent lane.*

as well as the physical representation of the cooperation coefficient c_i. In the IDM, the leaders' and followers' sets are defined based on the vehicle's current lateral position, y_k^i, leading to reactive behavior. In contrast, we propose to define the leader's and follower's sets based on the estimated lateral position at time-step H, $\tilde{y}_H^i$, as it follows

***Definition* 7.3.** *P-IDM: A set* $\mathcal{X}_i^l \subseteq \mathcal{X}$ *is the set of possible leaders for the vehicle i if* $\forall j \in [0, n-1], j \neq i : x_k^j < x_k^i$ *and* $\|\tilde{y}_H^i\| < c_i$.

Employing the predicted lateral position $\tilde{y}_H$ instead of the current lateral position y_k allows to elicit non-reactive behavior from the other vehicles. The leader for the vehicle i is defined as it follows

***Definition* 7.4.** *A vehicle* $j \in \mathcal{X}_i^l$ *is the leader of vehicle i if* $\forall m \in \mathcal{X}_i^l, m \neq j : \left\|x_k^j - x_k^i\right\| \leq \|x_k^m - x_k^i\|$.

To model mixed driving behavior, c_i is sampled from a uniform bounded distribution $c_i \sim \mathcal{U}([w_{\min}, w_{\max}])$ (defined in Section 7.5.3). $w_{\max}$ and $w_{\min}$ represents a maximum and minimum distance between the center of the current lane and the adjacent lane, as depicted in Figure 7.3, respectively. Moreover, the c_i values' range plays an essential role in the final policy's behavior by controlling the proportion of cooperative and non-cooperative vehicles encountered by the AV during training resulting in a more aggressive or conservative final policy.

Motion Plan Estimation

To enhance the IDM model with predictive driving behavior, we propose to condition the IDM on the beliefs of the other drivers' motion plans. Specifically, we assume that each vehicle on the main lane maintains an internal belief about the AV's motion plan (on the adjacent lane)[2]. To estimate the AV's motion plans, different prediction models can be employed (e.g., constant velocity model). Later, in Section 7.5.8, we investigate our method's performance for different prediction models.

Control Command Computation

For each time-step k and for each vehicle i, the acceleration control is computed depending on the vehicle's velocity v_k^i and current distance to the leader $\Delta x_k^i = \left\|(x_k^i, y_k^i) - (x_k^l, y_k^l)\right\|$:

$$u_k^{a,i} = a_{\max}\left[1 - \left(\frac{v_k^i}{v^*}\right)^4 - \left(\frac{s^*\left(v_k^i, \Delta v_k^i\right)}{\Delta x_k^i}\right)^2\right] \tag{7.9}$$

where s^* is the desired minimum gap, $a_{\max}$ the maximum acceleration, $\Delta v_k^i = v_k^i - v_k^l$ the i-th vehicle approach rate to the preceding vehicle, and v^* the desired velocity. Please note that we only do longitudinal control for the other vehicles on the main lane by employing Equation (7.9). For the AV, we employ a local optimization-based planner (Section 7.3.2) for steering and acceleration control.

7.5 Experiments

This section presents simulation results for two dense traffic scenarios (Section 7.5.2) considering different cooperation settings for the other vehicles (Section 7.5.3). First, we present qualitative (Section 7.5.6) and performance results (Section 7.5.7) of our approach against two baselines:

- DRL : state-of-the-art Deep Reinforcement Learning approach, SAC [46], learning a continuous policy controlling the AV's forward velocity.
- MPCC [143]: Model Predictive Contour Controller with a constant velocity reference.

After, we provide an ablation study analyzing our method's design choices (Section 7.5.8). All controller parameters were manually tuned to get the best possible performance.

7.5.1 Experimental Setup

Simulation results were carried out on an Intel Core i9, 32GB of RAM CPU @ 2.40GHz taking approximately 20 hours to train, approximately 20 million simulation steps. The non-linear

---

[2] For the Ramp Merging scenario (detailed in Sec. 7.5.2), the current lane corresponds to the main lane whereas the adjacent lane refers to the merge lane whereas for the Unprotected Left Turn scenario (detailed in Section 7.5.2), the current lane refers to the top lane and the adjacent lane corresponds to the bottom lane.

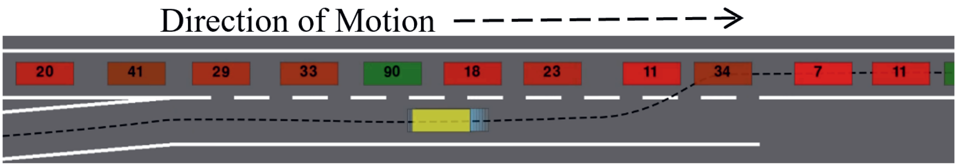
Table 7.1: Hyperparameters

Hyperparameter	Value
Planning Horizon	1.5 s
Number of Stages N	15
Number of parallel workers	7
Q neural network model	2 dense layers of 256
Policy neural network model	2 dense layers of 256
Activation units	Relu
Training batch size	2048
Discount factor	0.99
Optimizer	Adam
Initial entropy weight (α)	1.0
Target update (τ)	5×10^{-3}
Target entropy lower bound	-1.0
Target network update frequency	1
Learning rate	3×10^{-4}
Replay buffer size	10^6
$r_{\text{infeasible}}$	-1
$r_{\text{collision}}$	-300
r_{near}	-1.5
$\{q_c, q_l, q_v, q_u, q_\delta\}$	$\{0.1, 0.2, 1.0, 0.1, 0.1\}$
K	2
Timestep	0.1 s
Control cycle	0.2 s

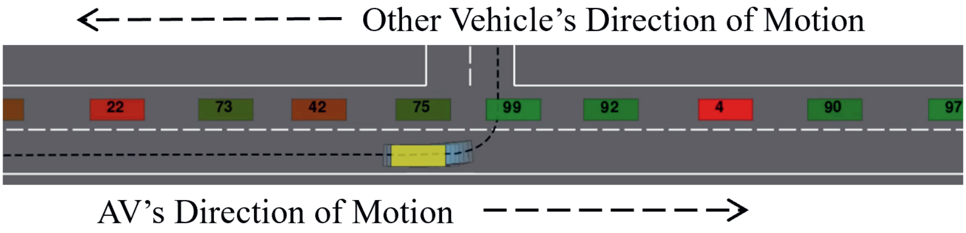
and non-convex MPCC problem of Equation (7.8) was solved using the ForcesPro [225] solver. Our simulation environment, P-IDM, builds on an open-source highway simulator [226] expanding it to incorporate complex interaction behavior. Hyperparameters values can be found in Table 7.1.

7.5.2 Driving Scenarios

We consider two densely populated driving scenarios: merging on a highway and unprotected left turn. The vehicles are modeled as rectangles with 5 m length and 2 m width. For each episode, the initial distance between the other vehicles is drawn from a uniform distribution ranging from [7, 10] m. Their initial and target velocities are sampled from a uniform distribution, $v_0^{0:n} \sim \mathcal{U}(3,4)$ m/s. This initial configuration prevents early collisions while ensuring no gaps of more than 2 meters [227], typical of dense traffic scenarios. These scenarios compel the AV to leverage other vehicles' cooperativeness while also exposing it to a myriad of critical scenarios for the final policy's performance.



(a) Ramp merging scenario. The AV on the main road, bottom lane, has to merge into the main top lane.



(b) Unprotected left-turn scenario: The AV on the main road, bottom lane, has to make a left-turn while avoiding collision with the other vehicles on the main road, top lane.

Figure 7.4: Evaluation environments: The AV is depicted in yellow and the reference path is depicted by the black dashed line. Each other vehicle is assigned with a color transitioning from red (i.e., non-cooperative) to green (i.e., cooperative). The number displayed by each vehicle represents its cooperation coefficient.

Ramp Merging

Figure 7.4a depicts an instance of the merging scenario. It comprises two lanes: the main lane and a merging lane. At the beginning of each episode, the main lane is populated with the other vehicles, moving from left to right. In contrast, the merge lane only includes the AV.

Unprotected Left Turn

Figure 7.4b illustrates the unprotected left turn scenario. It consists of two roads: the main road and the left road perpendicular to each other. The main road is populated with the other vehicles (on the top lane) and the AV (on the bottom lane). The other vehicles move from right to left on the main road, whereas the AV is initialized at the bottom lane of the main road, and its objective is to take an unprotected left turn onto the left road.

7.5.3 Evaluation Scenarios

We present simulation results considering different settings for the other vehicles' cooperation coefficient:

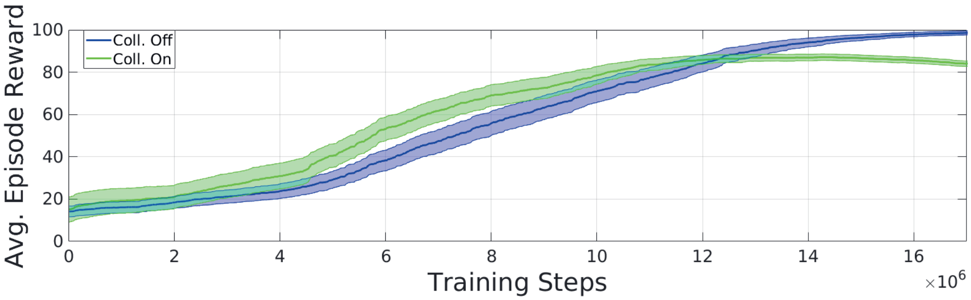
- Cooperative: In this scenario, most vehicles are cooperative ($c^i \sim \mathcal{U}(2,4)$ m), implying that as soon as the AV shows intentions of merging into the main lane, the other vehicle starts considering the AV as its new leader, leaving space for it to merge into the main lane. This evaluation scenario helps in assessing the merging speed of the policy.
- Non-Cooperative: This scenario comprises mostly non-cooperative vehicles ($c^i \sim \mathcal{U}(0,2)$ m), meaning that the other vehicles would not stop for the AV unless the AV's lateral horizon state is in the top lane. This scenario explicitly assesses the policy's aggressiveness. In these scenarios, the best option for the AV is to stop and wait for gaps and then merge in as quickly as possible.
- Mixed: This traffic scenario involves agents with varying degrees of cooperativeness ($c^i \sim \mathcal{U}(0,4)$ m), featuring a continuous transition from cooperative to non-cooperative vehicles. Here, the goal is to assess how differently the AV behaves with cooperative and non-cooperative vehicles.

During training, we consider a mixed setting for the other vehicles. Rule based methods such as IDM, MOBIL fail in highly dense traffic conditions and thus have not been included for evaluation purposes [162].

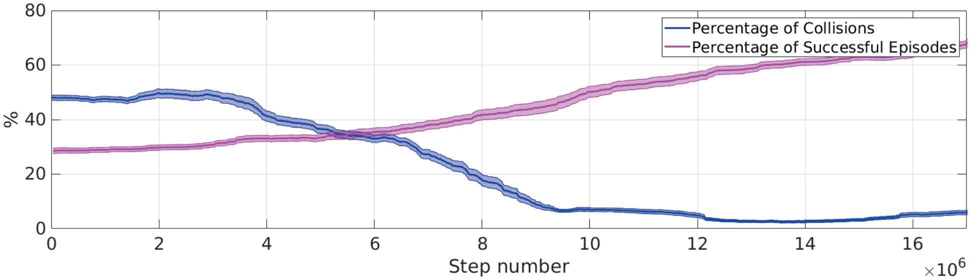
7.5.4 Evaluation Metrics

To evaluate our proposed method, we employ the following evaluation metrics:

- *Success Rate*: Percentage of successful episodes. An episode is deemed successful if the AV is able to merge on to the main highway or perform a left term without colliding and before timeout.
- *Collisions*: Percentage of episodes resulting in collision.
- *Timeout*: Percentage of episodes in which the AV did not reach the goal within the maximum specified time. This metric does not include those episodes that resulted in collision.
- *Time-to-goal*: Time in seconds for the AV to reach the goal position.



(a) Average reward during training when training with (green curve) and without (blue curve) collision constraints.



(b) Percentage of successful and failure episodes during training when training without collision constraints.

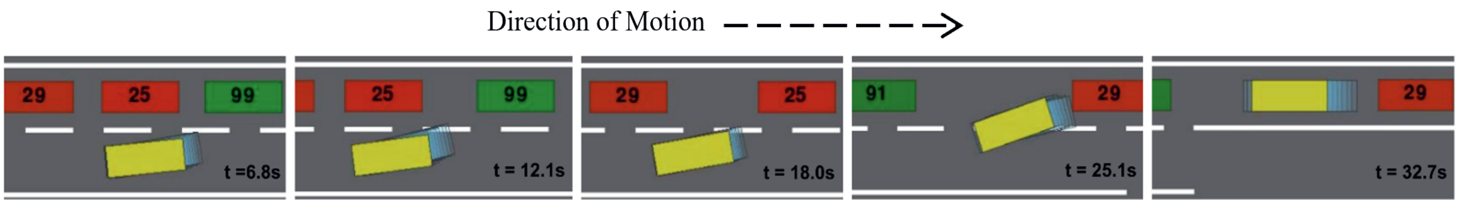
Figure 7.5: Training performance.

7.5.5 Training Procedure

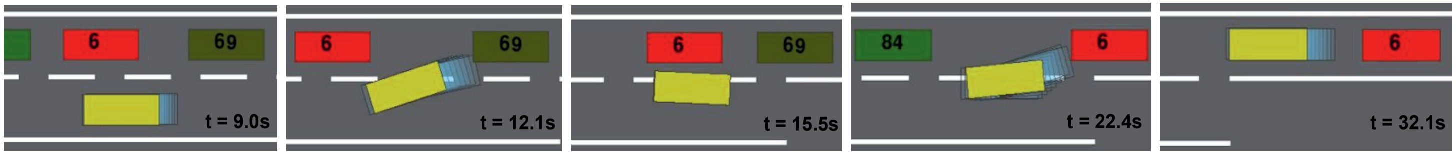
The interactive policy was trained considering a mixed setting of other vehicles following a P-IDM model with CV predictions. Figure 7.5 shows the performance of the learning policy during training. The top sub-plot (Figure 7.5a) shows the average reward evolution when training a policy with and without collision constraints. Training with collision constraints enables faster growth of the average rewards until 12×10^6 training steps. This phenomenon happens because the policy's task is simpler as the local controller overwrites the policy's actions that may lead to a collision. Nevertheless, employing collision constraints does not allow the AV to interact closely with the other vehicles. Hence, after the 12×10^6 training steps, the policy trained without collision constraints achieves a higher average reward. The bottom sub-plot (Figure 7.5b) shows the percentage of failure and collision episodes during training, demonstrating that the learning policy effectively decreases the percentage of collisions while increasing the rate of successful episodes throughout training.

7.5.6 Qualitative Results

Figure 7.6 presents visual results for our method for the merging scenarios and Figure 7.7 the left-turn scenarios. In Fig. 7.6a, the AV successfully merged onto the main lane by leveraging other vehicles' cooperativeness. In contrast, in Fig. 7.6b, we highlight a critical advantage of our framework: the ability to perform a collision avoidance maneuver when the guidance policy wrongly estimates the other vehicle's cooperativeness. In this episode, at 12.1 s, the AV initiates a merging maneuver. However, the non-cooperative vehicle does not allow it. The local planner aborts and starts a collision avoidance maneuver at 15.5 s, merging successfully later when encountering a cooperative vehicle at 22.4 s. Finally, Fig. 7.7a shows the AV performing an unprotected left-turn maneuver successfully. The presented qualitative results show that our proposed method enables the AV to safely and efficiently navigate in dense traffic scenarios.



(a) Successful merging maneuver: As the AV approaches the merging point, it tries to assess the reaction of its action on the vehicle titled "25" by inching closer to the main lane. The vehicle's non-cooperative behavior does not elicit a response typical of vehicles willing to yield, forcing the AV to stop. It tries the same with the vehicle titled "29" by creeping closer to the main lane but fails again. Finally, the merge is successful when a cooperative vehicle titled "91" emerges and gives way to the AV.



(b) Attempting to merge with a non-cooperative vehicle: In this episode, the guidance policy wrongfully estimates the other vehicle's non-cooperative nature, titled "6", compelling the AV to merge in front of the other agent. However, the obstacle avoidance constraint forces the AV to steer away from the other vehicle to avoid a collision. Finally, the AV merges in front of the cooperative vehicle titled "84".

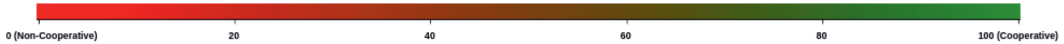
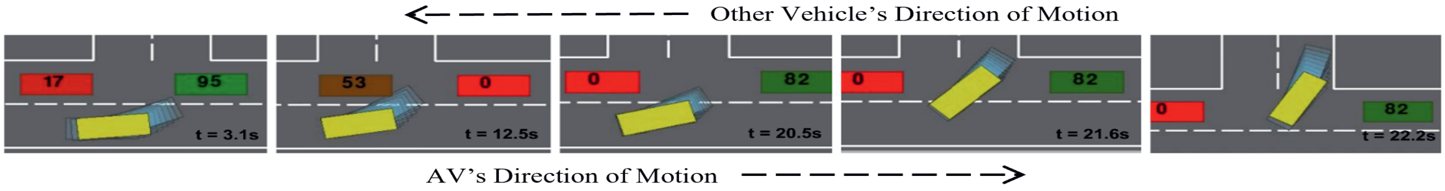


Figure 7.6: All the scenarios employ the P-IDM model (Section 7.4) to simulate the other vehicles. The AV is represented in yellow, whereas the future states, as computed by the MPCC, are plotted in light blue. Each other vehicle is assigned with a color transitioning from red (i.e., non-cooperative) to green (i.e., cooperative) to highlight the other vehicles' cooperativeness. The number displayed by each other vehicle represents its cooperation coefficient. All the numbers in between show a continuous transition from non-cooperative (0) to cooperative (100).



(a) Unprotected left-turn scenario: the AV approaches the center of the main lane to make the other vehicles yield. The first three vehicles it meets are non-cooperative and do not stop. When it meets a cooperative vehicle, titled "82", the AV behavior induces the other vehicle to yield allowing the AV to cross successfully.

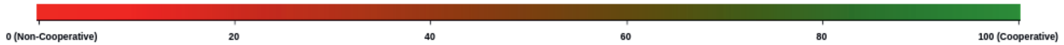


Figure 7.7: All the scenarios employ the P-IDM model (Section 7.4) to simulate the other vehicles. The AV is represented in yellow, whereas the future states, as computed by the MPCC, are plotted in light blue. Each other vehicle is assigned with a color transitioning from red (i.e., non-cooperative) to green (i.e., cooperative) to highlight the other vehicles' cooperativeness. The number displayed by each other vehicle represents its cooperation coefficient. All the numbers in between show a continuous transition from non-cooperative (0) to cooperative (100).

7.5.7 Quantitative Results

Aggregated results in Table 7.2 show that our method outperforms the baseline methods in terms of successful merges and number of collisions considering different settings for the other vehicles' behaviors (i.e., cooperative, mixed and, non-cooperative). The combined capability of interactive RL policy to implicitly embed inter-vehicle interactions into the velocity's policy and the safety provided by the collision avoidance constraints allows our method to succeed in all the environments. The optimization-based baseline (MPCC) shows poor performance for all settings, i.e., high collision rate. The reason is the lack of assimilation of inter-vehicle interactions into the policy and a tracking velocity reference error term in the cost function formulation that motivates the AV to keep the same velocity disregarding the nearby vehicles' cooperativeness. The DRL baseline achieves significantly higher performance, i.e., lower collision rate and a higher number of successful episodes. Nevertheless, it still leads to a significant number of collisions due to the lack of collision avoidance constraints to ensure safety when closely interacting with other vehicles. This demonstrates that employing collision constraints for navigation in dense traffic scenarios leads to superior performance over solely learning-based methods. In contrast, safety comes with the cost of larger average time-to-goal because the AV has to find the right time-window to merge.

Table 7.2: Statistic results for 1200 runs of proposed method (IntMPC) compared to baselines (MPCC [143] and DRL [46]) considering three different settings for the other vehicles (Section 7.5.3): percentage of success, collisions and timeout episodes.

	Cooperative			Mixed			Non Cooperative		
	Succ.(%)	Coll.(%)	Timeout(%)	Succ.(%)	Coll.(%)	Timeout(%)	Succ.(%)	Coll.(%)	Timeout(%)
MPCC [143]	78.0	11.0	10.0	62.0	22.0	16.0	26.0	57.0	19.0
RL [46]	86.0	3.0	11.0	69.0	2.0	29.0	31.0	5.0	64.0
Int-MPC	**86.0**	**0.0**	14.0	**70.0**	**0.0**	30.0	**36.0**	**0.0**	64.0

Table 7.3: Statistical results on the time-to-goal [s]. Only the episodes where all methods are successful are considered in the presented results. Bold values represent the results with statistical significance.

	Cooperative	Mixed	Non-cooperative
MPCC [143]	**34.7** ± **4.1**	**35.9** ± **6.9**	40.1 ± 5.8
DRL [46]	37.5 ± 7.8	37.9 ± 6.9	41.8 ± 7.4
IntMPC	37.6 ± 8.0	37.7 ± 6.0	41.0 ± 7.3

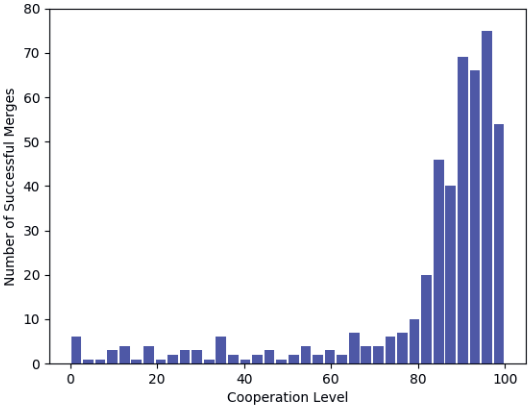


Figure 7.8: This figure provides a comprehensive analysis of the agents' cooperation level (0 - non cooperative, 100 - cooperative) in front of which the ego vehicle was able to merge successfully.

Table 7.3 presents statistical results of the *time-to-goal* for all methods. To evaluate the statistical significance, we performed pairwise Mann–Whitney U-tests between each method, considering a 95% confidence level. The results show statistical significance for the MPCC's results against the other methods for cooperative and mixed settings. In contrast, there is no statistical difference in terms of *time-to-goal* between the DRL and IntMPC. Similarly, between all methods in non-cooperative environments. The presented results show that employing collision avoidance constraints do not increase the average *time-to-goal* while improving safety. Moreover, in non-cooperative environments, all methods achieve comparable performance in terms of *time-to-goal*.

To demonstrate our policy's ability to leverage agents' cooperativeness explicitly, we evaluate 600 episodes in a mixed scenario where we track the other vehicle' cooperation level in front of which the AV performs a successful merging maneuver. Fig. 7.8 depicts a histogram illustrating the number of successful episodes per cooperation coefficient, demonstrating that our method mostly merges with cooperative vehicles. A small number of successful merges can be seen with non-cooperative vehicles as well. This behavior can be attributed to the random sampling of IDM parameters resulting in different agents' acceleration values. Thus, the agents might leave a gap big enough for the AV to merge

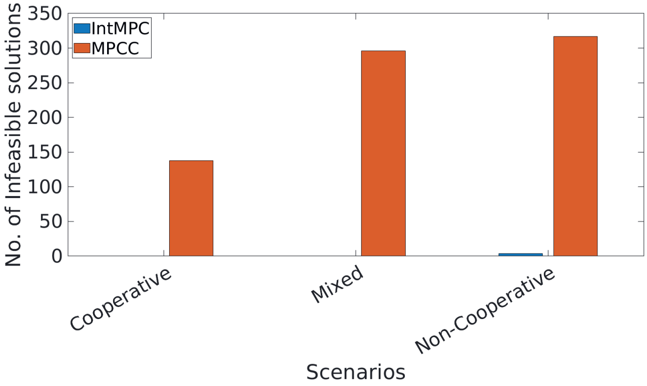


Figure 7.9: Number of infeasible solutions encountered by the solver for our method (IntMPC) versus the optimization-based baseline (MPCC).

onto the lane when moving from a standstill position.

Figure 7.9 presents the number of infeasible solutions for our method (IntMPC) and the MPCC baseline. To jointly train the RL policy with the local controller and penalize the state and action tuples resulting in the solver infeasibility, significantly reduces the number of infeasible solutions. Finally, in terms of computation performance, our policy's network has an average computation time of 1.35 ± 0.5 ms. To solve the IntMPC's optimization problem (Equation (7.8)) takes on average 3.0 ± 1.35 ms for all experiments. There was no statistical difference on the policy's and solver's computation times for the different settings of the other vehicles (e.g., cooperative, mixed and non-cooperative). These results demonstrate out method's real-time applicability.

7.5.8 Performance Analysis

This section investigates the impact of two critical design choices for our proposed approach: MPCC's parameters and using a different number of control cycles per RL policy query. Moreover, we evaluate our method's robustness to different prediction models used by the other vehicles to estimate the AV's motion plans. To finalize, we compare the risk-level that the AV takes with our method and the two planning baselines.

Local Controller Parameters

The MPCC's parameters (i.e., weights and velocity reference) highly influence the local planner's performance. Here, we study the two key components controlling the AV's interaction with the other vehicles: the velocity tracking weight (q_v) and the reference velocity (v_{ref}). Table 7.4 presents performance results for different q_v and v_{ref} values. Increasing the reference velocity combined with high q_v values generates more aggressive behavior and significantly reduces the timeout rate. However, it also increases the collision rate. In contrast, low q_v values weaken the influence of the velocity reference on the MPCC performance. The presented results demonstrate that fine-tuning the MPCC's weights and velocity reference is insufficient for safe and efficient navigation in dense traffic environments, supporting the need for an interaction-aware velocity reference. $q_v = 1.0$ and $v_{\text{ref}} = 2$ m/s lead to the best performance, i.e., higher success rate and lower collision and timeout rate. For the following experiments, we use $q_v = 1.0$ and a velocity reference of $v_{\text{ref}} = 2$ m/s for the MPCC baseline.

Table 7.4: Ablation study of the MPCC's parameters considering a mixed setting for the other vehicles.

Success (%) / Collision (%) / Timeout (%)			
	$q_v = 0.1$	$q_v = 1.0$	$q_v = 10.0$
$v_{\text{ref}} = 2$ m/s	58 / 26 / 16	**62 / 22 / 16**	58 / 34 / 8
$v_{\text{ref}} = 3$ m/s	48 / 26 / 26	62 / 29 / 9	55 / 42 / 3
$v_{\text{ref}} = 4$ m/s	56 / 25 / 19	57 / 35 / 8	53 / 46 / 1

Hyperparameter Selection

A key design choice of the proposed framework is the number of control cycles per policy query, denoted by K. For instance, for $K = 1$, we query the policy network for a new velocity reference for each control cycle, while for $K = 4$, we use the same queried velocity reference during 4 control cycles. Here, we study the impact on the learned policy's performance for $K = \{1, \ldots, 4\}$. During testing, all the policies are evaluated using $K = 1$. Table 7.5 summarizes the obtained performance results. The policy trained with $K = 2$ outperforms the other policies in terms of success and collision rate. The policy trained with $K = 1$ elicits an overly aggressive response from AV, evident from a high collision rate and a low timeout percentage. In contrast, higher K values lead the AV to exhibit an overly conservative behavior, thus, higher timeout percentage. This behavior can be attributed to the long duration for which the same action is applied after querying the interactive policy. For instance, using a large velocity reference value during many control cycles highly increases the collision likelihood at the merging point. This compels the RL algorithm to learn biased policy towards low-velocity references to avoid an impending collision resulting in an overly conservative behavior. Finally, the policy trained with $K = 2$ elicits a balanced response from the AV that is neither too conservative nor too aggressive, resulting in a high success rate and a low collision rate for all the scenarios.

Table 7.5: Sensitivity analysis of the hyperparameter K, i.e., number of control cycles per RL policy query, on the learned policy's performance. All policies were trained considering a mixed setting of other vehicles. Querying the RL policy for a new velocity for each two control cycles leads to the best performance (bold values).

	Cooperative			Mixed			Non Cooperative		
	Succ.(%)	Collision(%)	Timeout(%)	Succ.(%)	Coll.(%)	Timeout(%)	Succ.(%)	Coll.(%)	Timeout(%)
K = 1	80.0	0.0	20.0	70.0	0.0	30.0	33.0	0.0	67.0
K = 2	**88.0**	0.0	**12.0**	**72.0**	0.0	**28.0**	**37.0**	0.0	**63.0**
K = 3	71.5	0.0	28.5	46.75	0.0	53.25	5.5	0.0	94.5
K = 4	72.0	0.0	28.0	47.0	0.0	53.0	0.0	0.0	100.0

Simulation Environment

This work introduces an IDM variant enhancing the other vehicles with anticipatory behavior. Our proposed model (P-IDM in Section 7.4) relies on the assumption that the other vehicles can infer the AV's motion plans. Here, we evaluate the influence of the prediction model used to infer the AV's plans on our method's performance. We consider the following prediction models variants:

1. CV: Constant velocity (CV) model;
2. CVPath: Constant velocity (CV) model along the AV's reference path;
3. MPCC: MPCC plan (Equation (7.8)) assuming the AV's current velocity as the velocity reference, $v_{\text{ref}} = v_k$.

Moreover, we also evaluate our method's performance in reactive scenarios employing the IDM [224] to model the other vehicles' behaviors. The presented results in Table 7.6 demonstrate that our proposed approach is robust and generalizes well to environments with other vehicles exhibiting different behaviors. Employing the CV-Path prediction model results in highly cooperative behavior for other vehicles as shown by the high success rate. In contrast, the scenarios with vehicles following an IDM [224] represents the most challenging scenario.

Table 7.6: Analysis of the proposed method's performance when interacting with reactive (IDM [224]) and predictive vehicles (CV, CV-Path and MPCC).

	Cooperative			Mixed			Non Cooperative		
	Succ.(%)	Coll.(%)	Timeout(%)	Succ.(%)	Coll.(%)	Timeout(%)	Succ.(%)	Coll.(%)	Timeout(%)
React. Model									
IDM [224]	86.0	0.0	14.0	70.0	1.0	29.0	36.0	0.0	64.0
Pred. Model									
CV	88.0	0.0	12.0	72.0	0.0	28.0	37.0	0.0	63.0
CV-Path	98.0	0.0	2.0	88.0	0.0	12.0	37.0	0.0	63.0
MPCC	89.0	0.0	11.0	76.0	0.0	24.0	39.0	0.0	61.0

Table 7.7: Risk-level analysis

Time of Closest Encounter [s] / Distance of Closest Encounter [m]			
	Cooperative	Mixed	Non-cooperative
MPCC [143]	9.3 / 3.44	13.6 / 3.13	13.5 / 3.07
RL [46]	26.5 / 3.45	45.6 / 3.15	43.6 / 3.15
IntMPC	25.6 / 3.51	46.0 / 3.20	43.1 / 3.17

Risk-level Analysis

Table 7.7 compares the risk-level that the AV takes using our approach against the baseline methods for two risk metrics as proposed in [133]: Time of Closest Encounter (TCE) and the Distance-of-Closest-Encounter (DCE). DCE models how close the AV gets to the other vehicles meaning that lower DCE represents higher risk. TCE models the risk time-dependency, assuming that risk events further away in time have a lower probability of occurrence. Hence, the larger TCE, the lower the risk. The presented results show that our method incurs the lowest risk.

7.5.9 Discussion

The presented performance and ablation results demonstrate that our approach improves performance and safety significantly compared to pure learning or optimization baselines. Our approach enables the AV to exploit the interaction effects in the other agents to efficiently and safely perform different driving maneuvers by employing RL to learn an interaction-aware velocity reference directly fed into the MPCC's cost function. Nevertheless, the sensitivity analysis results presented in Table 7.6 show some performance degradation when evaluating our approach in scenarios containing agents following different policies from those used in the training scenarios. This effect is due to the *sim-to-real* gap inherent to RL methods [228], and it can be exacerbated when evaluating our approach in real environments.

7.6 Conclusions

This section introduced an interaction-aware policy for guiding a local optimization planner through dense traffic scenarios. We proposed to model the interaction policy as a velocity reference and employed DRL methods to learn a policy maximizing long-term rewards by exploiting the interaction effects. Then, a MPCC is used to generate control commands satisfying collision and kino-dynamic constraints when a feasible solution is found. Learning an interaction-aware velocity reference policy enhances the MPCC planner with interactive behavior necessary to safely and efficiently navigate in dense traffic. The presented results show that our method outperforms solely learning-based and optimization-based planners in terms of collisions, successful maneuvers, and fewer deadlocks in cooperative, mixed, and non-cooperative scenarios.

8

Conclusions and Future Work

8.1 Conclusions

This thesis presents algorithms that enable safe autonomous navigation of mobile robots in environments populated with humans and other robots. The first goal of this thesis is to develop a motion planning algorithm endowing computing predictive motions plans online for mobile robots and autonomous vehicles in unstructured, dynamic environments. Computing predictive motion plans requires planning over a prediction horizon and, therefore, requires a world model that estimates its evolution. Consequently, the second goal is to create human trajectory prediction models accounting for interaction, environment constraints, and multi-modality. In dynamic environments, generating trajectories online requires planning over a limited horizon and, consequently, computing locally optimal solutions. Hence, the third objective is to develop a method for providing long-term guidance to a local planner. Lastly, to navigate efficiently in dense traffic scenarios, it is crucial to interact with the other agents. Thus, the last goal of this thesis is to create a method to enhance local trajectory planners with interactive behavior to exploit the interaction with other agents in dense traffic scenarios.

Chapter 4 presented a Local Model Predictive Contouring Control method (LMPCC) enabling safe autonomous navigation in dynamic and unstructured environments. Here, we tackled two main problems. Firstly, previous works modeling dynamic obstacles' space as circles and ellipsoids used an incorrect bound and collisions could still occur when used for motion planning. Therefore, we proposed a closed-form bound to conservatively approximate the Minkowski sum of a circle and an ellipse which can be formulated as a non-linear inequality constraint into the optimization problem to avoid collisions with dynamic obstacles. Secondly, using the same bound to model the complex structure of indoor environments would require many computationally expensive constraints and result in an overly conservative approximation of the static obstacles' space. To account for static obstacles, we proposed an algorithm to compute a polyhedral approximation of the collision-free area around the robot. The polyhedral approximation consists of a set of linear constraints reducing the computational demands compared to the ellipsoid-circle constraints used for dynamic collision avoidance. We implemented the proposed method in an autonomous mobile robot and an autonomous vehicle with onboard localization and obstacle detection, which was shown to avoid walking humans at a maximum speed of 1.5 m/s. Additionally, our proposed Model Predictive Contouring Control scheme was shown to perform in real-time achieving computation times under 50 ms for planning horizons below 3 s. It was demonstrated that the proposed motion planner significantly improved navigation performance in terms of safety, reducing by 44% and 35% the number of episodes resulting in collisions compared to learning-based and traditional methods, respectively. The primary reason for this result is that our approach plans N-steps ahead, endowing the robot with anticipatory behavior while the baseline approaches are purely reactive. Moreover, in terms of efficiency, the average traveled distance was reduced by approximately 15% and 23% compared to learning-based and reactive methods, respectively.

Chapter 5 tackles the problem of predicting multi-modal trajectory from a single network query. State-of-the-art approaches employed random sampling from the latent space distribution to predict multi-modal trajectories (i.e., to predict multiple possible trajectories).

However, these approaches suffer from mode-collapse (i.e., model can only predict one or a very small subset of possible outcomes), generating very similar trajectories. Moreover, most methods relied on generative adversarial neural networks (GANs) as the model, which is very difficult and unstable to train. Therefore, Chapter 5 introduced a novel Variational Recurrent Neural Network (VRNN) architecture for one-shot multi-modal trajectory prediction. The presented architecture learns the parameters of a Gaussian Mixture Model to account for uncertainty and multi-modality. It incorporates information about pedestrian dynamics, static obstacles, and other interacting pedestrians to predict trajectories accounting for interaction and environment constraints. Additionally, by employing a variational approach, the proposed model has stable training and faster convergence than GAN-based methods. The qualitative results presented in Chapter 5 show that the Social-VRNN predictions avoid static obstacles and account for the interactions with surrounding pedestrians. Moreover, qualitative results demonstrate that the proposed model generates more distinct trajectories than the state-of-the-art prediction model [93]. The performance results demonstrate that the proposed model reduces the average displacement error (ADE) by 9 % and the final displacement error (FDE) in 25% compared to two baseline prediction models [92, 93].

In Chapter 6, the addressed problem is two-fold. Firstly, local trajectory optimization methods scale poorly with the number of agents and require considering limited planning horizon to generate a solution online, leading to catastrophic failures if the final goal position is outside the planning horizon. Secondly, deep RL-based methods overcome the scalability issues and enable long-term planning but do not necessarily satisfy hard constraints. Thus, Chapter 6 introduced the Goal-Oriented Model Predictive Control (GO-MPC) framework enhancing a state-of-art online optimization-based planner (MPC) with a learned subgoal policy providing global guidance. The MPC ensures generating trajectories satisfying collision and dynamic constraints. The DRL enables learning a policy accounting for the long-term planning horizon and interaction with other agents. During training, performance results demonstrate that the proposed framework achieves on average over 20% higher average episode reward compared to solely optimization-based and learning-based [17] approaches. Qualitative results show that the learned policy exhibits different behaviors depending on the cooperativeness of other agents. Additionally, the results show that the proposed approach enables decentralized and communication-free multi-robot collision avoidance. The performance results show a reduction in the percentage of failure episodes (i.e., collisions and deadlocks) between 13% to 22% compared to the MPC baseline without guidance, depending on the number of agents ranging from six to ten. Compared to the DRL [17] and DRL-2 baseline, our method reduces the percentage of failure episodes between 17% to 29% and between 6% to 13%, respectively. Similarly, when considering a multi-robot setting, the performance test results show a reduction in the percentage of failure episodes between 13% to 33% compared to the MPC baseline, between 16% to 18% compared to the DRL baseline [17] and between 17% to 26% compared to the DRL-2 baseline [17]. In terms of efficiency, the GO-MPC only outperforms the DRL-2 baseline [17] presenting on average larger travel time and traveled distance than the MPC and DRL baselines. The first explanation behind these results is that the GO-MPC has to behave more conservatively to bring the percentage of collisions close to zero. Secondly, DRL learns a policy or guidance behavior that allows solving simple and complex scenarios simultaneously. Consequently, the qualitative results show that the GO-MPC can show over-conservative behavior in simple scenarios (e.g., low number of agents).

Finally, in Chapter 7 we tackled the problem of autonomous navigation in dense traffic scenarios (e.g., merging on highways and unprotected left-turn maneuvers). Autonomous vehicles must exploit interaction with other cars to enable them to navigate in dense traffic successfully. Therefore it is crucial to endow AVs with interactive behavior. Hence, we proposed the interactive Model Predictive Controller (IntMPC) utilizing an interaction-aware policy providing a velocity reference to the local planner. The learned policy allows leveraging the vehicles' interaction effects to create free-space areas. The LMPCC computes the steering and throttle commands for the AV respecting collision and dynamic constraints. In contrast to the GO-MPC approach proposed in Chapter 6, in the IntMPC framework the DRL policy provides a velocity reference as global guidance information allowing to control the interaction with the other vehicles. The qualitative results show that the learned policy enhances the LMPCC with interactive behavior to pro-actively merge in dense traffic. The performance results show that the IntMPC improves the navigation performance and safety substantially. Concerning optimization-based baselines, the proposed method increased over 8% the success rate in cooperative, mixed, and non-cooperative settings. As compared to DRL baseline, the proposed approach reduced by 3%, 2%, and 5% the percentage of collisions in cooperative, mixed, and non-cooperative environments, respectively. Moreover, it increased the success rate by 5% in non-cooperative scenarios compared to the DRL baseline. Finally, the risk-analysis results show that, on average, the IntMPC presents larger Time to Collision (TTC), Time of Closest Encounter (TCE), and Distance of Closest Encounter compare to the baselines. Therefore, incurring lower collision risk.

Overall, these contributions address the key challenges in safe and efficient robot navigation among humans and other robots. However, there are still several remaining issues before we can safely deploy autonomous robots everywhere. Toward that goal, possible directions of future work are described below.

8.2 Future Work

This thesis contributed to motion planning, prediction, and decision-making algorithms, enabling safe and efficient navigation among humans. Nevertheless, many open challenges still need to be addressed before broadly releasing autonomous systems (e.g., mobile robots and autonomous vehicles) in human environments. Hereafter, we recommend several future research directions to extend this work for interaction-aware motion planning, learning for socially-aware guidance, and precise control.

8.2.1 Interaction-Aware Motion Planning

This work proposed to learn offline a policy controlling the interaction with the other agents through the MPC's cost function. However, such an approach is prone to generalization issues as the agents' behavior might be significantly different in reality compared to simulation. Moreover, it requires to assume that the interactions among the agents are decoupled during the planning phase which is not realistic.

The collision avoidance problem is inherently a game. Each autonomous agent's actions are coupled with the others. Game-theoretic approaches considering a feedback information structure have shown that accounting for this action coupling between all agents in the motion planning phase allows discovering strategies exploiting the actions of the others [229]. Yet, this comes with the cost of the so-called "curse of dimensionality," limiting game-theoretic approaches to scenarios with a small number of agents.

It is critical to develop algorithms enabling online computation of game solutions. Future works may explore supervised learning to learn offline a joint model for all agent dynamics. Then, employing differentiable MPC [230] is a promising solution to find approximate solutions to the Nash equilibrium online.

8.2.2 Learning Guidance Policies

This thesis proposed two methods to solve the curse of dimensionality issues inherent to local optimization methods when navigating in cluttered environments (Chapter 6) or accounting for interaction (Chapter 7) by learning a policy providing global guidance in the local planner's cost function. However, further research on learning global guidance policies is still required. Firstly, to apply the GO-MPC in real scenarios, it is necessary to expand the method to account for static obstacles. Secondly, further research is needed to improve the GO-MPC performance in terms of travel time and distance. As pointed out in Section 8.1, the GO-MPC can show highly conservative behavior in simple scenarios (e.g., swapping position with another agent), which is one of the results behind the poor efficiency performance (i.e., average travel time and traveled distance). Moreover, the qualitative results presented in Chapter 6 show that the learned behavior does not resemble the pedestrians behavior. Thus, further research to enable learning socially compliant guidance policies is necessary. Techniques such as Generative Adversarial Imitation Learning [231] and Adversarial Inverse Reinforcement Learning [232] are promising solutions. Yet, these approaches are sensitive to hyper-parameter tuning and challenging to train, requiring further research.

8.2.3 Constraints-Aware Learning

The state-of-the-art supervised and reinforcement learning methods used in this thesis can learn policies capable of solving complex tasks from high-dimensional information. Yet, these methods cannot learn policies respecting the kinematic and dynamic constraints of the system. For instance, DRL enables learning guidance policies that, combined with constrained optimization methods (e.g., the GO-MPC approach proposed in this thesis), enable robots to navigate cluttered environments. However, the generated guidance information does not necessarily respect dynamic or collision constraints relying on the local planner to handle those.

Most state-of-the-art learning methods neglect the geometric knowledge that we have about the system. Thus, to advance learning methods incorporating such information is critical to enable learning policies for control. Two recent promising works in this direction are Implicit Behavior Cloning [233] and physics-informed neural networks [234, 235]. But they have not been yet explored for autonomous navigation.

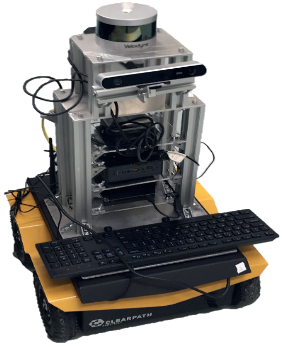
8.2.4 Continual Learning

The learning methods proposed in this thesis rely on large amounts of data gathered offline through large datasets or simulation environments to learn the model's parameters. However, offline learning is prone to generalization issues, leading to catastrophic failure in real or novel environments. In addition, state-of-the-art learning algorithms have fixed performance not improving over time.

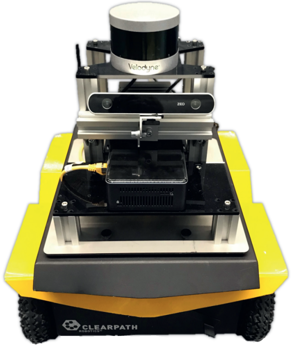
Current autonomous robotic systems have access to online data streams (e.g., sensor data, algorithms outputs, etc.), providing valuable information from the environment where they are deployed. Hence, it is essential to develop learning algorithms exploiting online data while retaining past knowledge to improve the model's (e.g., prediction and decision-making) performance in unseen environments and enable the next generation of robots to adapt to non-stationary environments.

A

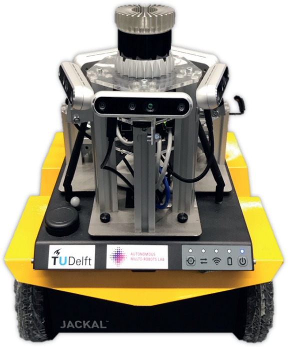
The Jackal



(a) Jackal v1.



(b) Jackal v2.



(c) Jackal v3.

The algorithms presented in this thesis have been fully implemented on-board and tested on the Jackal platform from Clearpath. The Jackal's design has been evolving throughout the development of this research giving birth to the three Jackal generations presented in Appendix A. All the designs are equipped with a Velodyne LiDAR (VLP-16) with 16 laser beams used for localization and pedestrian detection, an IMU, and an Intel i5 CPU@2.6GHz. The first and second versions, Jackal v1 and Jackal v2, were also equipped with a ZED stereo camera for pedestrian detection. Additionally, both platforms used an Nvidia Xavier GPU board and an Intel NUC i7 CPU@3.2GHz as extra processing power units. In the Jackal v3 version, the additional processing power units (i.e., Intel NUC and Nvidia Xavier) have been replaced by a Dell laptop, simplifying the hardware complexity, reducing power consumption from the robot's batteries, and easier software development and debugging. Moreover, the stereo cameras have been replaced by the Intel RealSense, allowing to reduce the robot's computational demands as the Intel cameras have stereo on-board processing. The Jackal v3 setup has five Intel RealSense cameras enabling 360 degrees of perception range.

B

The Toyota Prius

The LMPCC method presented in Chapter 4 has been tested on the TU Delft Toyota Prius to avoid a pedestrian, as depicted in Figure B.1. The car has a PC mounted on board running Ubuntu 18.04.1 LTS with an Intel(R) Core(TM) i7-6900K CPU at 3.20GHz with 64GB of RAM. In addition, the PC contains two Titan X (Pascal) GPUs for stereo matching. For localization, the vehicle uses a high-precision dual GPS mounted on top. Finally, there the car uses one stereo camera for VRU detection.

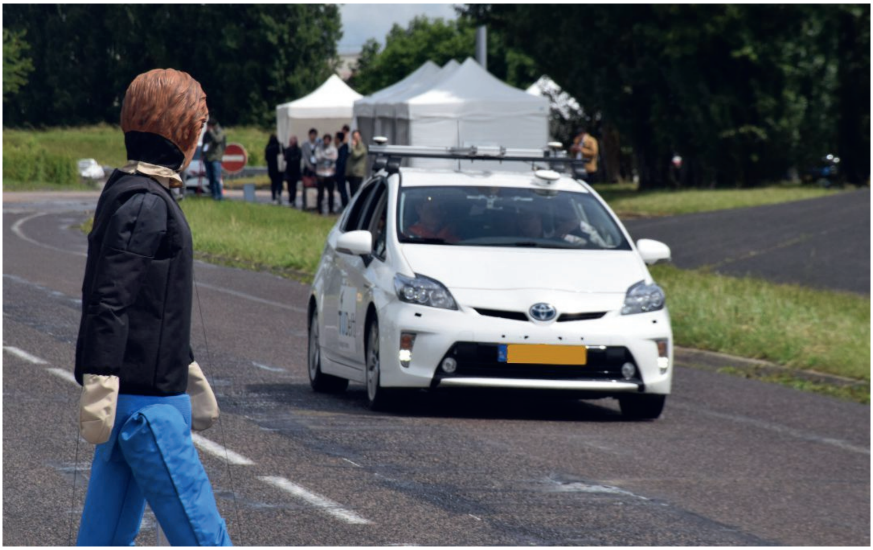


Figure B.1: The TU Delft Toyota Prius

Bibliography

[1] P. A. Hancock, Illah Nourbakhsh, and Jack Stewart. On the future of transportation in an era of automated and autonomous vehicles. *Proceedings of the National Academy of Sciences*, 116(16):7684–7691, 2019.

[2] Andrew J. Hawkins. Motional's robotaxis will be fully driverless in las vegas by 2023, Nov 2021.

[3] Andrew J. Hawkins. Starship technologies' delivery robots are coming to more college campuses this fall, Aug 2021.

[4] Vice President of Products at Amazon Charlie Tritschler. Meet astro, a home robot unlike any other, Sep 2021.

[5] Alphabet's waymo and gm's cruise get california dmv approval to run commercial autonomous car services. https://www.cnbc.com/2021/09/30/waymo-and-cruise-get-california-dmv-approval-to-run-driverless-cars.html. Accessed: 2021-11-04.

[6] Javier Alonso-Mora, Paul Beardsley, and Roland Siegwart. Cooperative collision avoidance for nonholonomic robots. *IEEE Transactions on Robotics*, 34(2):404–420, 2018.

[7] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[8] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and Decision-Making for Autonomous Vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):060117–105157, 2018.

[9] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[10] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[11] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[12] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1928–1935. IEEE, 2008.

[13] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807. IEEE, 1993.

[14] Wilko Schwarting, Alyssa Pierson, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences of the United States of America*, 116(50):2492–24978, 2019.

[15] Jaime F Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S Shankar Sastry, and Anca D Dragan. Hierarchical game-theoretic planning for autonomous vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9590–9596. IEEE, 2019.

[16] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. Lucidgames: Online unscented inverse dynamic games for adaptive trajectory prediction and planning. *IEEE Robotics and Automation Letters*, 6(3):5485–5492, 2021.

[17] M. Everett, Y. F. Chen, and J. P. How. Collision avoidance in pedestrian-rich environments with deep reinforcement learning. *IEEE Access*, 9:10357–10377, 2021.

[18] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6015–6022. IEEE, 2019.

[19] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308, 2020.

[20] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *International Conference on Robotics and Automation*, 2010.

[21] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems.* Cambridge University Press, 2017.

[22] Hai Zhu and Javier Alonso-Mora. Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783, 2019.

[23] Wilko Schwarting, Javier Alonso-Mora, Liam Pauli, Sertac Karaman, and Daniela Rus. Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1928–1935. IEEE, 2017.

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

[25] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[26] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[27] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29:4743–4751, 2016.

[28] Szilárd Aradi. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[29] Steven M La Valle. Motion planning. *IEEE Robotics & Automation Magazine*, 18(2):108–118, 2011.

[30] Jiirg P Keller and Brian DO Anderson. A new approach to the discretization of continuous-time controllers. In *1990 American Control Conference*, pages 1127–1132. IEEE, 1990.

[31] Lei Tai, Jingwei Zhang, Ming Liu, Joschka Boedecker, and Wolfram Burgard. A survey of deep network solutions for learning control in robotics: From reinforcement to imitation. *arXiv preprint arXiv:1612.07139*, 2016.

[32] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[33] 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.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[35] Christopher M Bishop. Pattern recognition. *Machine learning*, 128(9), 2006.

[36] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.

[37] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *International Conference on Machine Learning*, volume 2, page 2. Citeseer, 2014.

[38] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[39] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018.

[40] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[42] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[43] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Philip S. Yu. Learning multiple tasks with multilinear relationship networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1593–1602, Red Hook, NY, USA, 2017. Curran Associates Inc.

[44] Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[46] Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

[47] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[48] Gonzalo Ferrer, Anais Garrell, and Alberto Sanfeliu. Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1688–1694. IEEE, 2013.

[49] Javier Alonso-Mora, Paul Beardsley, and Roland Siegwart. Cooperative Collision Avoidance for Nonholonomic Robots. *IEEE Transactions on Robotics*, 34(2):404–420, April 2018.

[50] Jan Marian Maciejowski. *Predictive control: with constraints.* Pearson education, 2002.

[51] Francesco Borrelli, Paolo Falcone, Tamas Keviczky, Jahan Asgari, and Davor Hrovat. MPC-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3(2-4):265–291, 2005.

[52] C. M. Kang, S. Lee, and Chung Choo Chung. On-road path generation and control for waypoints tracking. *IEEE Intelligent Transportation Systems Magazine*, 9(3):36–45, 2017.

[53] Thomas M. Howard, Colin J. Green, and Alonzo Kelly. Receding horizon model-predictive control for mobile robot navigation of intricate paths. In Andrew Howard, Karl Iagnemma, and Alonzo Kelly, editors, *Field and Service Robotics*, pages 69–78, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[54] David Fridovich-Keil, Sylvia L. Herbert, Jaime F. Fisac, Sampada Deglurkar, and Claire J. Tomlin. Planning, fast and slow: A framework for adaptive real-time safe trajectory planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 387–394, 2018.

[55] Jaime F Fisac, Andrea Bajcsy, Sylvia L Herbert, David Fridovich-Keil, Steven Wang, Claire J Tomlin, and Anca D Dragan. Probabilistically safe robot planning with confidence-based human predictions. *Robotics: Science and Systems (RSS)*, 2018.

[56] D. Lam, C. Manzie, and M. Good. Model predictive contouring control. In *49th IEEE Conference on Decision and Control (CDC)*, 2010.

[57] Matthew Brown, Joseph Funke, Stephen Erlien, and J Christian Gerdes. Safe driving envelopes for path tracking in autonomous vehicles. *Control Engineering Practice*, 61:307–316, 2017.

[58] Wilko Schwarting, Javier Alonso-Mora, Liam Paull, Sertac Karaman, and Daniela Rus. Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model. *IEEE Transactions on Intelligent Transportation Systems*, 19(9):2994–3008, 2018.

[59] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

[60] Pete Trautman. Sparse interacting gaussian processes: Efficiency and optimality theorems of autonomous crowd navigation. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017.

[61] Björn Lütjens, Michael Everett, and Jonathan P. How. Safe reinforcement learning with model uncertainty estimates. *International Conference on Robotics and Automation (ICRA)*, 2019.

[62] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6252–6259. IEEE, 2018.

[63] Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauss, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, Christoph Gustav Keller, et al. Making Bertha Drive–An Autonomous Journey on a Historic Route. *IEEE Intelligent Transportation Systems Magazine*, 6(2):8–20, 2014.

[64] Alberto López Rosado, Stanley Chien, Lingxi Li, Qiang Yi, Yaobin Chen, and Rini Sherony. Certainty and critical speed for decision making in tests of pedestrian automatic emergency braking systems. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1358–1370, 2017.

[65] Christoph G Keller, Thao Dang, Hans Fritz, Armin Joos, Clemens Rabe, and Dariu M Gavrila. Active pedestrian safety by automatic braking and evasive steering. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1292–1304, 2011.

[66] Thao Dang, Jens Desens, Uwe Franke, Dariu Gavrila, Lorenz Schäfers, and Walter Ziegler. Steering and evasion assist. In *Handbook of intelligent vehicles*, pages 759–782. Springer, 2012.

[67] Sebastian Köhler, Brian Schreiner, Steffen Ronalter, Konrad Doll, Ulrich Brunsmann, and Klaus Zindler. Autonomous evasive maneuvers triggered by infrastructure-based detection of pedestrian intentions. In *IV Symposium*, pages 519–526, 2013.

[68] Thomas Grußner, Lutz Bürkle, and Claus Marberger. Erweiterung Aktiver Fußgängerschutzsysteme Durch Eine Fahrerinitierte Ausweichunterstützung. In *Uni-DAS e.V. Workshop Fahrerassistenzsysteme*, pages 19–28, 2015.

[69] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

[70] Sertac Karaman and Emilio Frazzoli. Incremental Sampling-based Algorithms for Optimal Motion Planning. *International Journal of Robotics Research*, 5 2010.

[71] Steven Lavalle. *Planning Algorithms*. Cambridge University Press, Cambridge, 2006.

[72] Yoshiaki Kuwata, Gaston Fiore, Justin Teo, Emilio Frazzoli, and Jonathan How. Motion planning for urban driving using RRT. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 1681–1686. IEEE, 9 2008.

[73] Chris Urmson and Reid Simmons. Approaches for Heuristically Biasing RRT Growth. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2, pages 1178–1183, 2003.

[74] Emilio Frazzoli, Munther A. Dahleh, and Eric Feron. Real-Time Motion Planning for Agile Autonomous Vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[75] Kwangjin Yang. An efficient Spline-based RRT path planner for non-holonomic robots in cluttered environments. *2013 International Conference on Unmanned Aircraft Systems, ICUAS 2013 - Conference Proceedings*, pages 288–297, 2013.

[76] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[77] Sertac Karaman and Emilio Frazzoli. Sampling-based optimal motion planning for non-holonomic dynamical systems. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5041–5047, 2013.

[78] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime Motion Planning using the RRT*. *IEEE Conference on Robotics and Automation*, 2011.

[79] Yoshiaki Kuwata, Justin Teo, Gaston Fiore, Sertac Karaman, Emilio Frazzoli, and Jonathan P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

[80] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics, 1995.

[81] Sujeong Kim, Stephen J. Guy, Wenxi Liu, David Wilkie, Rynson W.H. Lau, Ming C. Lin, and Dinesh Manocha. BRVO: Predicting pedestrian trajectories using velocity-space reasoning. *International Journal of Robotics Research*, 34(2):201–217, 2015.

[82] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 797–803. IEEE, 2010.

[83] Stefan Becker, Ronny Hug, Wolfgang Hübner, and Michael Arens. An evaluation of trajectory prediction approaches and notes on the trajnet benchmark. *ArXiv*, abs/1805.07663, 2018.

[84] Hao Xue, Du Q Huynh, and Mark Reynolds. Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1186–1194. IEEE, 2018.

[85] Mark Pfeiffer, Giuseppe Paolo, Hannes Sommer, Juan Nieto, Rol Siegwart, and Cesar Cadena. A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[86] 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.

[87] Irtiza Hasan, Francesco Setti, Theodore Tsesmelis, Alessio Del Bue, Fabio Galasso, and Marco Cristani. Mx-lstm: mixing tracklets and vislets to jointly forecast trajectories and head poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6067–6076, 2018.

[88] Federico Bartoli, Giuseppe Lisanti, Lamberto Ballan, and Alberto Del Bimbo. Context-aware trajectory prediction. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1941–1946. IEEE, 2018.

[89] Kaiping Xu, Zheng Qin, Guolong Wang, Kai Huang, Shuxiong Ye, and Huidi Zhang. Collision-free lstm for human trajectory prediction. In *International Conference on Multimedia Modeling*, pages 106–116. Springer, 2018.

[90] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. *Computer Graphics Forum*, 26, 2007.

[91] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

[92] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019.

[93] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. Social ways: Learning multimodal distributions of pedestrian trajectories with gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[94] Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7144–7153, 2019.

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

[96] Rohan Chandra, Tianrui Guan, Srujan Panuganti, Trisha Mittal, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms. *IEEE Robotics and Automation Letters*, 5(3):4882–4890, 2020.

[97] Stuart Eiffert, Kunming Li, Mao Shan, Stewart Worrall, Salah Sukkarieh, and Eduardo Nebot. Probabilistic crowd gan: Multimodal pedestrian trajectory prediction using a graph vehicle-pedestrian attention network. *IEEE Robotics and Automation Letters*, 5(4):5026–5033, 2020.

[98] Tim Salzmann, B. Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *ECCV*, 2020.

[99] Weiming Zhi, Ransalu Senanayake, Lionel Ott, and Fabio Ramos. Spatiotemporal learning of directional uncertainty in urban environments with kernel recurrent mixture density networks. *IEEE Robotics and Automation Letters*, 4(4):4306–4313, 2019.

[100] Weiming Zhi, Lionel Ott, and Fabio Ramos. Kernel trajectory maps for multi-modal probabilistic motion prediction. In *Conference on Robot Learning*, pages 1405–1414, 2020.

[101] B. Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2375–2384, 2019.

[102] Yu Fan Chen, Shih-Yuan Liu, Miao Liu, Justin Miller, and Jonathan P How. Motion planning with diffusion maps. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.

[103] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*. Springer, 2011.

[104] Jur P. van den Berg, Jamie Snape, Stephen J. Guy, and Dinesh Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. *2011 IEEE International Conference on Robotics and Automation*, pages 3475–3482, 2011.

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

[106] Christoforos I Mavrogiannis and Ross A Knepper. Multi-agent path topology in support of socially competent navigation planning. *The International Journal of Robotics Research*, 38(2-3):338–356, 2019.

[107] Christoforos I Mavrogiannis, Wil B Thomason, and Ross A Knepper. Social momentum: A framework for legible navigation in dynamic multi-agent environments. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 361–369, 2018.

[108] Pete Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015.

[109] Beomjoon Kim and Joelle Pineau. Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics*, 8(1):51–66, 2016.

[110] Anirudh Vemula, Katharina Muelling, and Jean Oh. Modeling cooperative navigation in dense human crowds. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1685–1692. IEEE, 2017.

[111] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

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

[113] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research*, 39:856 – 892, 2020.

[114] Lei Tai, Jingwei Zhang, Ming Liu, and Wolfram Burgard. Socially compliant navigation through raw depth inputs with generative adversarial imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1111–1117. IEEE, 2018.

[115] Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[116] Nicolas Mansard, Andrea DelPrete, Mathieu Geisert, Steve Tonneau, and Olivier Stasse. Using a memory of motion to efficiently warm-start a nonlinear predictive controller. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2986–2993. IEEE, 2018.

[117] Guillaume Bellegarda and Katie Byl. Combining benefits from trajectory optimization and deep reinforcement learning, 2019.

[118] Mario Zanon and Sébastien Gros. Safe reinforcement learninge using robust mpc. *IEEE Transactions on Automatic Control*, 2020.

[119] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning, 2017.

[120] Mingyuan Zhong, Mikala Johnson, Yuval Tassa, Tom Erez, and Emanuel Todorov. Value function approximation and model predictive control. In *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pages 100–107. IEEE, 2013.

[121] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

[122] Farbod Farshidian, David Hoeller, and Marco Hutter. Deep value model predictive control. *arXiv preprint arXiv:1910.03358*, 2019.

[123] Napat Karnchanachari, Miguel I. Valls, David Hoeller, and Marco Hutter. Practical reinforcement learning for mpc: Learning from sparse objectives in under an hour on a real robot, 2020.

[124] Damien Ernst, Mevludin Glavic, Florin Capitanescu, and Louis Wehenkel. Reinforcement learning versus model predictive control: a comparison on a power system problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):517–529, 2008.

[125] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[126] Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. In *Advances in neural information processing systems*, 2013.

[127] Igor Mordatch and Emo Todorov. Combining the benefits of function approximation and trajectory optimization. In *Robotics: Science and Systems*, volume 4, 2014.

[128] Zhang-Wei Hong, Joni Pajarinen, and Jan Peters. Model-based lookahead reinforcement learning, 2019.

[129] Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *ArXiv*, abs/1906.08649, 2020.

[130] Colin Greatwood and Arthur G Richards. Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control. *Autonomous Robots*, 43(7):1681–1693, 2019.

[131] Bruno Brito, Michael Everett, Jonathan P How, and Javier Alonso-Mora. Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments. *IEEE Robotics and Automation Letters*, 6(3):4616–4623, 2021.

[132] Joseph Lubars, Harsh Gupta, Adnan Raja, R. Srikant, Liyun Li, and Xinzhou Wu. Combining reinforcement learning with model predictive control for on-ramp merging. *CoRR*, abs/2011.08484, 2020.

[133] Julian Eggert. Predictive risk estimation for intelligent adas functions. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 711–718, 2014.

[134] Florian Damerow and Julian Eggert. Predictive risk maps. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 703–710. IEEE, 2014.

[135] Christopher R. Baker and John M. Dolan. Traffic interaction in the urban challenge: Putting boss on its best behavior. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 1752–1758, 2008.

[136] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, and C. Geyer et Al. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[137] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, and B. Huhnke et Al. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 25(1):569–597, 2008.

[138] Haoyu Bai, Shaojun Cai, Nan Ye, David Hsu, and Wee Sun Lee. Intention-aware online pomdp planning for autonomous driving in a crowd. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 454–460, 2015.

[139] W Liu, S Kim, S Pendleton, and M H Ang. Situation-aware decision making for autonomous driving on urban road using online POMDP. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1126–1133, 6 2015.

[140] Constantin Hubmann, Marvin Becker, Daniel Althoff, David Lenz, and Christoph Stiller. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1671–1678, 2017.

[141] Constantin Hubmann, Jens Schulz, Gavin Xu, Daniel Althoff, and Christoph Stiller. A Belief State Planner for Interactive Merge Maneuvers in Congested Traffic. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-Novem(November 2018):1617–1624, 2018.

[142] Bingyu Zhou, Wilko Schwarting, Daniela Rus, and Javier Alonso-Mora. Joint multi-policy behavior estimation and receding-horizon trajectory planning for automated urban driving. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2388–2394, 2018.

[143] Laura Ferranti, Bruno Brito, E Pool, Y Zheng, Ronald M Ensing, Riender Happee, B Shyrokau, Julian FP Kooij, Javier Alonso-Mora, and Dariu M Gavrila. Safevru: A research platform for the interaction of self-driving vehicles with vulnerable road users. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1660–1666. IEEE, 2019.

[144] Jae Sung Park, Chonhyon Park, and Dinesh Manocha. I-planner: Intention-aware motion planning using learning-based human motion prediction. *The International Journal of Robotics Research*, 38(1):23–39, 2019.

[145] Bruno Brito, Hai Zhu, Wei Pan, and Javier Alonso-Mora. Social-vrnn: One-shot multi-modal trajectory prediction for interacting pedestrians. *Conference on Robot Learning*, 2020.

[146] Sangjae Bae, Dhruv Saxena, Alireza Nakhaei, Chiho Choi, Kikuo Fujimura, and Scott Moura. Cooperation-aware lane change maneuver in dense traffic based on model predictive control with recurrent neural network. In *2020 American Control Conference (ACC)*, pages 1209–1216. IEEE, 2020.

[147] Boris Ivanovic, Amine Elhafsi, Guy Rosman, Adrien Gaidon, and Marco Pavone. Mats: An interpretable trajectory forecasting representation for planning and control. *Conference on Robot Learning*, 2020.

[148] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for autonomous cars that leverage effects on human actions. *Robotics: Science and Systems*, 12, 2016.

[149] Changxi You, Jianbo Lu, Dimitar Filev, and Panagiotis Tsiotras. Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Robotics and Autonomous Systems*, 114:1–18, 2019.

[150] Colin F. Camerer, Teck Hua Ho, and Juin Kuan Chong. A cognitive hierarchy model of games. *Quarterly Journal of Economics*, 119(3):861–898, 2004.

[151] Mario Garzón and Anne Spalanzani. Game theoretic decision making for autonomous vehicles' merge manoeuvre in high traffic scenarios. *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, pages 3448–3453, 2019.

[152] Maxime Bouton, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Mykel J Kochenderfer. Reinforcement learning with iterative reasoning for merging in dense traffic. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[153] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211. IEEE, 2017.

[154] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700. IEEE, 2018.

[155] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3399–3406, 2018.

[156] Alexander Amini, Wilko Schwarting, Guy Rosman, Brandon Araki, Sertac Karaman, and Daniela Rus. Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 568–575. IEEE, 2018.

[157] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and A. Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *ArXiv*, abs/1710.05465, 2017.

[158] M. Bouton, A. Nakhaei, K. Fujimura, and Mykel J. Kochenderfer. Cooperation-aware reinforcement learning for merging in dense traffic. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3441–3447, 2019.

[159] Pin Wang, Chingyao Chan, and Arnaud de La Fortelle. A reinforcement learning based approach for automated lane change maneuvers. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1379–1384, 2018.

[160] Mustafa Mukadam, Akansel Cosgun, Alireza Nakhaei, and Kikuo Fujimura. Tactical decision making for lane changing with deep reinforcement learning. 2017.

[161] Tommy Tram, Anton Jansson, Robin Grönberg, Mohammad Ali, and Jonas Sjöberg. Learning negotiating behavior between cars in intersections using deep q-learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3169–3174. IEEE, 2018.

[162] Dhruv Mauria Saxena, Sangjae Bae, Alireza Nakhaei, Kikuo Fujimura, and Maxim Likhachev. Driving in dense traffic with model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5385–5392. IEEE, 2020.

[163] Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *AAAI*, 2018.

[164] Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019.

[165] Richard Cheng, Mohammad Javad Khojasteh, A. Ames, and Joel W. Burdick. Safe multi-agent interaction through robust control barrier functions with learned uncertainties. *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 777–783, 2020.

[166] Maxime Bouton, Jesper Karlsson, Alireza Nakhaei, Kikuo Fujimura, Mykel J. Kochenderfer, and Jana Tumova. Reinforcement learning with probabilistic guarantees for autonomous driving, 2019.

[167] Tommy Tram, Ivo Batkovic, Mohammad Ali, and Jonas Sjöberg. Learning when to drive in intersections by combining reinforcement learning and model predictive control. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3263–3268, 2019.

[168] Alexander Domahidi and Juan Jerez. FORCES Professional. embotech GmbH (http://embotech.com/FORCES-Pro), July 2014.

[169] Jan Maciejowski. *Predictive Control With Constraints*. 01 2002.

[170] 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.

[171] Shinichi Nakajima, Kazuho Watanabe, and Masashi Sugiyama. *Variational Bayesian Learning Theory*. Cambridge University Press, 2019.

[172] Hao Dong, Hao Dong, Zihan Ding, Shanghang Zhang, and Chang. *Deep Reinforcement Learning*. Springer, 2020.

[173] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[174] Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3059. IEEE, 2018.

[175] Greg Welch and Gary Bishop. An introduction to kalman filter. In *SIGGRAPH 2001*, 1995.

[176] Jianyu Chen, Wei Zhan, and Masayoshi Tomizuka. Constrained iterative LQR for on-road autonomous driving motion planning. In *20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[177] Alexei Yu Uteshev and Marina V Yashina. Metric problems for quadrics in multidimensional space. *Journal of Symbolic Computation*, 68:287–315, 2015.

[178] Timm Linder, Stefan Breuers, B. Leibe, and Kai Oliver Arras. On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5512–5519, 2016.

[179] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 2011.

[180] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots.* The MIT Press, 2 edition, 2011.

[181] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 2014.

[182] Zhenji Lu, Barys Shyrokau, Boulaid Boulkroune, Sebastiaan van Aalst, and Riender Happee. Performance benchmark of state-of-the-art lateral path-following controllers. In *15th IEEE International Workshop on AMC*, pages 541–546, 2018.

[183] Tass International. Delft-tyre 6.2.

[184] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora. Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robotics and Automation Letters*, 4(4):4459–4466, 2019.

[185] Jiahao Lin, Hai Zhu, and Javier Alonso-Mora. Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2682–2688. IEEE, 2020.

[186] Parth Kothari, Sven Kreiss, and Alexandre Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *arXiv preprint arXiv:2007.03639*, 2020.

[187] Björn Lötjens, Michael Everett, and Jonathan P How. Safe reinforcement learning with model uncertainty estimates. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8662–8668. IEEE, 2019.

[188] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[189] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

[190] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268. IEEE, 2009.

[191] Safdar Zaman, Wolfgang Slany, and Gerald Steinbauer. Ros-based mapping, localization and autonomous navigation using a pioneer 3-dx robot and their relevant issues. In *2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC)*, pages 1–5. IEEE, 2011.

[192] G. Q. Huang, A. B. Rad, and Y. K. Wong. Online slam in dynamic environments. In *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pages 262–267, 2005.

[193] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[194] Christopher M Bishop. Mixture density networks. Technical report, Citeseer, 1994.

[195] Alex Graves. Generating sequences with recurrent neural networks, 2013.

[196] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018.

[197] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.

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

[199] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1(2), 2015.

[200] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

[201] Trajnet++ (A Trajectory Forecasting Challenge).

[202] Bruno Brito, Boaz Floor, Laura Ferranti, and Javier Alonso-Mora. Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robotics and Automation Letters*, 4(4):4459–4466, 2019.

[203] Steven M LaValle. *Planning algorithms.* Cambridge university press, 2006.

[204] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios. *ArXiv*, abs/1808.03841, 2018.

[205] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935. IEEE, 2008.

[206] Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019.

[207] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[208] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines, 2018.

[209] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[210] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *European Conference on Computer Vision*, pages 414–430. Springer, 2020.

[211] Simon Ulbrich, Simon Grossjohann, Christian Appelt, Kai Homeier, Jens Rieken, and Markus Maurer. Structuring cooperative behavior planning implementations for automated driving. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2159–2165. IEEE, 2015.

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

[213] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A Review of Motion Planning Techniques for Automated Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2016.

[214] Pete Trautman. Sparse interacting Gaussian processes: Efficiency and optimality theorems of autonomous crowd navigation. *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, 2018-Janua:327–334, 2018.

[215] Arkady Zgonnikov, David A. Abbink, and Gustav Markkula. Should i stay or should i go? evidence accumulation drives decision making in human drivers. 2020.

[216] Tomer Toledo, Haris N Koutsopoulos, and Moshe E Ben-Akiva. Modeling integrated lane-changing behavior. *Transportation Research Record*, 1857(1):30–38, 2003.

[217] Florian Marczak, Winnie Daamen, and Christine Buisson. Key variables of merging behaviour: empirical comparison between two sites and assessment of gap acceptance theory. *Procedia-Social and Behavioral Sciences*, 80:678–697, 2013.

[218] Gary A Davis and Tait Swenson. Field study of gap acceptance by left-turning drivers. *Transportation Research Record*, 1899(1):71–75, 2004.

[219] Yang Yu. Towards sample efficient reinforcement learning. In *IJCAI*, pages 5739–5743, 2018.

[220] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[221] Wilko Schwarting, Javier Alonso-Mora, Liam Paull, Sertac Karaman, and Daniela Rus. Safe Nonlinear Trajectory Generation for Parallel Autonomy with a Dynamic Vehicle Model. *IEEE Transactions on Intelligent Transportation Systems*, 19(9):2994–3008, 2018.

[222] B Berg, B Brito, J Alonso-Mora, and M Alirezaei. Curvature Aware Motion Planning with Closed-Loop Rapidly-exploring Random Trees. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021.

[223] Bruno Brito, Boaz Floor, Laura Ferranti, and Javier Alonso-Mora. Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments. *IEEE Robotics and Automation Letters*, 4(4):4459–4466, 2019.

[224] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[225] Andrea Zanelli, Alexander Domahidi, J Jerez, and Manfred Morari. Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 93(1):13–29, 2020.

[226] Edouard Leurent. An Environment for Autonomous Driving Decision-Making, 2018.

[227] Daiheng Ni, John D Leonard, Chaoqun Jia, and Jianqiang Wang. Vehicle longitudinal control and traffic stream modeling. *Transportation Science*, 50(3):1016–1031, 2016.

[228] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 734–743. PMLR, 2018.

[229] David Fridovich-Keil, Ellis Ratner, Lasse Peters, Anca D Dragan, and Claire J Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1475–1481. IEEE, 2020.

[230] Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J. Zico Kolter. Differentiable mpc for end-to-end planning and control. In *NeurIPS*, 2018.

[231] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NIPS*, 2016.

[232] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[233] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. *Conference on Robot Learning (CoRL)*, November 2021.

[234] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[235] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Curriculum vitæ

Bruno Brito was born in September 1990 in Porto, Portugal. He received the M.Sc. (2013) degree from the Faculty of Engineering of the University of Porto. Between 2014 and 2016, he was a trainee at the European Space Agency (ESA) in the Guidance, Navigation, and Control section. After, he was a Research Associate, between 2016 and 2018, in the Fraunhofer Institute for Manufacturing Engineering and Automation.

In January 2018, he became a Ph.D. candidate at the Department of Cognitive Robotics, Delft University of Technology, Delft, the Netherlands. In his Ph.D. project, he worked on motion planning algorithms for autonomous navigation among humans under the supervision of Dr. Javier Alonso-Mora. During 2020, he collaborated with the Aerospace Controls Laboratory (ACL), MIT, Boston, United States, researching on learning global guidance policies, together with Dr. Michael Everett and Prof. Jonathan P. How.

His research interests include robot motion planning, deep learning, reinforcement learning, model predictive control, and learning-based motion planning.

List of publications

Referred journals

- R. Pérez-Dattari*, B. Brito*, O. de Groot, J. Kober and J. Alonso-Mora, "Visually-guided motion planning for autonomous driving from interactive demonstrations," in *Engineering Applications of Artificial Intelligence*, 116 (2022): 105277.
- B. Brito, A. Agarwal and J. Alonso-Mora, "Learning Interaction-aware Guidance Policies for Motion Planning in Dense Traffic Scenarios," in *IEEE Transactions on Intelligent Transportation Systems*, doi: 10.1109/TITS.2022.3160936.
- H. Zhu, B. Brito, and J. Alonso-Mora, "Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware voronoi cells," in *Autonomous Robots*, 46(2): 401-420 (2022).
- L. Knoedler, C. Salmi, H. Zhu, B. Brito, and J. Alonso-Mora, "Improving pedestrian prediction models with self-supervised continual learning," in *IEEE Robotics and Automation Letters*, 7(2): 4781-4788 (2022).
- B. Brito, M. Everett, J. How and J. Alonso-Mora, "Where to go Next: Learning a Subgoal Recommendation Policy for Navigation in Dynamic Environment," in *IEEE Robotics and Automation Letters*, 6(3):4616-4623, July. 2021.
- O. de Groot, B. Brito, L. Ferranti, D. Gavrila and J. Alonso-Mora, "Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments," in *IEEE Robotics and Automation Letters*, 6(3):5389-5396, July. 2021.
- H. Zhu*, F.M. Claramunt*, B. Brito, and J. Alonso-Mora, "Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments," in *IEEE Robotics and Automation Letters*, 6(2):2256-2263, Apr. 2021.
- B. Brito, B. Floor, L. Ferranti and J. Alonso-Mora, "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments," in *IEEE Robotics and Automation Letters*, 4(4):4459-4466, Oct. 2019.

---

* indicates equal contributions.

Referred conference proceedings

- M. Lodel, B. Brito, Á. Serra-Gómez, L. Ferranti, R. Babuska and J. Alonso-Mora, "Where to Look Next: Learning Viewpoint Recommendations for Informative Trajectory Planning," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 4466-4472.

- J. de Vries, E. Trevisan, J. van der Toorn, T. Das, B. Brito, and J. Alonso-Mora, "Regulations Aware Motion Planning for Autonomous Surface Vessels in Urban Canals," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 3291-3297.

- B. van den Berg, B. Brito, M. Alirezaei, and J. Alonso-Mora, "Curvature Aware Motion Planning with Closed-Loop Rapidly-exploring Random Trees,"in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2021.

- M. Spahn, B. Brito, and J. Alonso-Mora, "Coupled Mobile Manipulation via Trajectory Optimization with Free Space Decomposition," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, June 2021.

- B. Brito, H. Zhu, W. Pan, and J. Alonso-Mora, "Social-vrnn: One-shot multi-modal trajectory prediction for interacting pedestrians," in *Conference on Robot Learning (CoRL)*, Nov. 2020.

- Á. Serra-Gómez, B. Brito, H. Zhu, J.J Chung, and J. Alonso-Mora, "With whom to communicate: Learning efficient communication for multi-robot collision avoidance," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, Oct. 2020.

- L. Ferranti*, B. Brito*, E. Pool, Y. Zheng, R. Ensing, R. Happee, B. Shyrokau, J. Kooij, J. Alonso-Mora and D. Gavrila, "SafeVRU: A research platform for the interaction of self-driving vehicles with vulnerable road users," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2019.

TU Delft