



Delft University of Technology

Model Predictive Path Integral Control for Interaction-Rich Local Motion Planning in Dynamic Environments

Trevisan, E.

DOI

[10.4233/uuid:e7343dd8-cab4-4679-92a6-84a67430af54](https://doi.org/10.4233/uuid:e7343dd8-cab4-4679-92a6-84a67430af54)

Publication date

2025

Document Version

Final published version

Citation (APA)

Trevisan, E. (2025). *Model Predictive Path Integral Control for Interaction-Rich Local Motion Planning in Dynamic Environments*. [Dissertation (TU Delft), Delft University of Technology].
<https://doi.org/10.4233/uuid:e7343dd8-cab4-4679-92a6-84a67430af54>

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.

Model Predictive Path Integral Control for Interaction-Rich Local Motion Planning in Dynamic Environments

Model Predictive Path Integral Control for Interaction-Rich Local Motion Planning in 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 of Doctorates,

to be defended publicly on

Wednesday, 26 March 2025 at 10.00

by

Elia TREVISAN

This dissertation has been approved by the promotor.

promotors: Dr. J. Alonso-Mora, Prof. dr. R. Babuška

Composition of the doctoral committee:

Rector Magnificus,
Dr. J. Alonso-Mora,
Prof. dr. R. Babuška,

Chairperson
Delft University of Technology, *promotor*
Delft University of Technology, *promotor*

Independent Members:

Prof. dr. R. R. Negenborn,
Prof. dr. N. van de Wouw,
Prof. dr. C. Pradalier,
Dr. A. Liniger,
Prof. dr. M. Wisse,

Delft University of Technology
Eindhoven University of Technology
GeorgiaTech Lorraine, France
Robotics and AI Institute, Switzerland
Delft University of Technology, *reserve member*

The work in the thesis has been funded by the project “Sustainable Transportation and Logistics over Water: Electrification, Automation and Optimization (TRiLOGy)” of the Netherlands Organization for Scientific Research (NWO), domain Science (ENW), and the Amsterdam Institute for Advanced Metropolitan Solutions (AMS) in the Netherlands.



Keywords: Motion Planning, Model Predictive Path Integral Control, Interaction-Aware Planning, Autonomous Vehicles, Marine Robotics

Printed by: Ridderprint | www.ridderprint.nl

Cover: Sketched with OpenAI's DALL-E, hand-drawn by Elia Trevisan

Style: TU Delft House Style, with modifications by Moritz Beller
<https://github.com/Inventitech/phd-thesis-template>

The author set this thesis in $\text{\LaTeX}$ using the Libertinus and Inconsolata fonts.

ISBN 978-94-6518-027-4

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

The absolute truth is that if you don't know what you want, you won't get it.

Andrew S. Grove

Contents

Summary	xi
Samenvatting	xiii
Acknowledgments	xv
1 Introduction	1
1.1 Limitations of Current Approaches.	3
1.2 Research Questions	3
1.3 Contributions	4
1.4 Outline.	5
2 Preliminaries	7
2.1 Path Integral Control.	8
2.1.1 Monte Carlo Sampling	9
2.2 Information-Theoretic Approach	10
2.2.1 Iterative Importance Sampling	13
3 Interaction-Aware Local Motion Planning	15
3.1 Introduction	16
3.1.1 Related Work.	17
3.1.2 Contributions	17
3.2 Preliminaries.	18
3.2.1 MPPI Algorithm	18
3.2.2 Vessel State and Dynamics	19
3.2.3 Global Planning	19
3.3 Decentralized Interaction-Aware MPPI.	19
3.3.1 Approach and Architecture	19
3.3.2 Two-stage Sample Evaluation	21
3.3.3 Cost Formulation.	21
3.3.4 Local Goal Prediction	23
3.4 Experiments	24
3.4.1 Comparison with Optimization-based MPC	24
3.4.2 Navigation in Crowded Environments	25
3.4.3 Navigation among Human-piloted Vessels	27
3.4.4 Computational Complexity.	27
3.5 Conclusions	29
4 Learning-Based Predictions for Autonomous Surface Vessels	31
4.1 Introduction	32
4.1.1 Related Work.	32

4.1.2	Contribution	34
4.2	Interaction-Aware MPPI	34
4.3	Predicting Goal Positions.	36
4.3.1	Interaction-Aware Trajectory Prediction Method.	36
4.3.2	Artificial Dataset	37
4.3.3	Model Training and Adaptation	37
4.3.4	Local Goal Extraction	38
4.4	Experiments	39
4.4.1	Prediction Accuracy	39
4.4.2	Interaction-Aware Motion Planning with Predictions.	40
4.4.3	Decoupled Prediction and Planning	42
4.5	Conclusions	44
5	Informing the Motion Planner with Ancillary Controllers	47
5.1	Introduction	48
5.1.1	Previous Work	48
5.1.2	Contributions	49
5.2	Preliminaries.	50
5.3	Proposed Approach	52
5.3.1	Biased-MPPI	52
5.3.2	Sampling from Ancillary Controllers	53
5.3.3	Autotuning the Inverse Temperature	54
5.4	Illustrative Experiment.	54
5.4.1	Swing-up and Tracking	56
5.5	Simulated Motion Planning Experiments.	57
5.5.1	Solving an Intersection.	57
5.5.2	Interaction-Aware Planning with Four Vessels	58
5.6	Real-World Motion Planning Experiment.	62
5.7	Conclusions	63
6	Applications to Contact-Rich Robot Manipulation	65
6.1	Introduction	66
6.1.1	Related Work.	67
6.1.2	Contributions	68
6.2	Sampling-based MPC via Parallelizable Physics Simulations	68
6.2.1	Background Theory on MPPI.	68
6.2.2	Proposed Algorithm	69
6.2.3	Exploiting the Physics Simulator Features	71
6.3	Experiments	71
6.3.1	Motion Planning and Collision Avoidance	72
6.3.2	Prehensile Manipulation with Whole-Body Control	72
6.3.3	Non-Prehensile Manipulation	74
6.3.4	Real-World Experiments	79
6.4	Discussion	81
6.5	Conclusions	81

7 Conclusion and Discussion	83
7.1 Conclusion	84
7.1.1 Interaction-Aware MPPI	84
7.1.2 Learning-Based Trajectory Predictions	84
7.1.3 Biased-MPPI	85
7.1.4 Isaac-MPPI	85
7.2 Discussion	85
7.2.1 MPPI or MPC	85
7.2.2 Future Work on Sampling-Based MPC	86
7.2.3 Considerations and Vision on Unmanned Vessels	87
Bibliography	91
Glossary	104
Curriculum Vitæ	105
List of Publications	107

Summary

In urban environments like Amsterdam, where road congestion and stress on infrastructure are critical issues, the city's waterways remain underutilized for transportation. Autonomous Surface Vessels (ASVs), making waterway transport cheaper and less labor intensive, present a potential solution by reducing transportation costs and alleviating road traffic. Although companies are developing ASVs, current solutions are mainly limited to larger, less congested waterways, with further innovation needed for denser urban areas.

The challenges in autonomous navigation for ASVs in urban canals stem from the complex nonlinear dynamics of vessels, which require long-term planning and rapid responses to environmental changes. Urban waterways are narrow and unstructured, with loosely defined navigation rules that can lead to discontinuities in the motion planner's cost function. Typically, motion planners rely on predictions of other agents' movements and plan around them, resulting in no interaction awareness. This approach can lead to the freezing robot problem in dense environments, where the ASV halts, deeming all the space unsafe. A local motion planner must address these challenges by ensuring long-horizon interaction-aware motions, rule adherence, and real-time planning.

An emerging approach in autonomous navigation is a sampling-based Model Predictive Control (MPC) strategy known as Model Predictive Path Integral control (MPPI). This algorithm approximates the optimal control sequence by sampling from a continuous input distribution. Thanks to its gradient-free nature, MPPI only requires collision checking to avoid collision and allows discontinuous cost functions. Additionally, its computation speed remains largely unaffected by the complexity of the robots' nonlinear dynamics, enabling longer planning horizons. While this approach has grown in popularity in the motion planning community, its application in dynamic environments with multiple interacting agents remains relatively unexplored.

Building on the state-of-the-art in MPPI, this thesis first introduces an Interaction-Aware Model Predictive Path Integral (IA-MPPI) controller tailored for motion planning in crowded urban canals. While conventional planners passively react to other agents, IA-MPPI actively predicts and plans cooperatively in real-time, addressing the freezing robot problem and handling discontinuous navigation rules. The decentralized, communication-free architecture assumes agent cooperation and employs a two-stage sample evaluation to enhance efficiency. This approach manages nonlinear dynamics, exact obstacle shapes, and longer planning horizons, demonstrating robustness and efficiency compared to state-of-the-art MPC in multi-agent environments.

While IA-MPPI uses a constant velocity model for predicting other agents' hidden goals, Chapter 4 of this thesis introduces a learning-based trajectory prediction model trained

on realistic artificial data to improve accuracy in crowded environments. By extracting the agents' goals from predicted trajectories and integrating this information into the IA-MPPI controller, the planner's performance increases significantly in environments with tight interactions. Moreover, this approach outperforms treating the predicted trajectory as occupied space, leading to safer navigation in dense urban canals.

One of MPPI's main drawbacks is that it struggles in highly dynamic environments because it usually only samples around a previous plan. When rapid changes occur, such as strong disturbances or an obstacle unexpectedly cutting off the path, the previous plan becomes invalid, and all the generated samples may lead to collisions. To address this, Chapter 5 introduces Biased-MPPI, incorporating multiple classic and learning-based ancillary controllers into the sampling distribution. While the mathematical derivations show that this introduces a bias, this significantly improves reactivity, safety, and robustness to local minima. Additionally, this approach reduces the required samples, making the controller more efficient regarding computational demand.

Lastly, Chapter 6 leverages the gradient-free nature of MPPI and applies it to a different domain involving contact-rich manipulation tasks, which are notoriously difficult for model-based planning. With a parallelizable GPU-based physics engine like IsaacGym, MPPI can efficiently roll out hundreds of input sequences in parallel, faster than in real-time, to approximate optimal control. This approach eliminates the need for explicit modeling of contact dynamics by exploiting the models embedded in the simulator, making it particularly suited for manipulation tasks like pushing and picking with both prehensile and non-prehensile manipulators. Experiments with real robots demonstrate the effectiveness of this method in handling discontinuous, contact-heavy environments.

Overall, this thesis explores key aspects of motion planning, particularly for ASVs, with extensions to ground robots and mobile manipulators. It advances the MPPI framework for autonomous navigation by adapting it for multi-agent systems and introducing biases through classic and learning-based ancillary controllers. Additionally, the thesis compares MPPI to MPC in navigation experiments and to optimization fabrics in manipulation tasks. The work presented promotes the use of MPPI, especially in domains where complex nonlinear dynamics and discontinuous costs complicate the use of real-time optimization-based MPC.

Samenvatting

In stedelijke omgevingen zoals Amsterdam, waar verkeersopstoppingen en druk op de infrastructuur kritieke problemen zijn, blijven de waterwegen van de stad onderbenut voor transport. Autonomous Surface Vessels (ASVs), die waterwegtransport goedkoper en minder arbeidsintensief maken, bieden een potentiële oplossing door de transportkosten te verlagen en het wegverkeer te ontlasten. Hoewel bedrijven ASVs ontwikkelen, zijn de huidige oplossingen voornamelijk beperkt tot grotere, minder drukke waterwegen, en is verdere innovatie nodig voor dichtbevolkte stedelijke gebieden.

De uitdagingen van autonome navigatie voor ASVs in stedelijke grachten komen voort uit de complexe niet-lineaire dynamica van schepen, die lange termijnplanning en snelle reacties op omgevingsveranderingen vereisen. Stedelijke waterwegen zijn smal en ongestructureerd, met vaak gedefinieerde navigatieregels die leiden tot discontinuïteiten in de bewegingsplanning wanneer ze als kostenfunctie worden toegevoegd. Gewoonlijk vertrouwen bewegingsplanners op voorspellingen van de bewegingen van other agents en plannen daaromheen, wat resulteert in een gebrek aan interactie-bewustzijn. Deze aanpak kan in drukke omgevingen leiden tot het 'freezing robot'-probleem, waarbij de ASV stopt omdat alle ruimte als onveilig wordt beschouwd. Een lokale bewegingsplanner moet deze uitdagingen aanpakken door door interactie-bewuste bewegingen over een lange planningshorizon te waarborgen, evenals naleving van regels en real-time planning.

Een opkomende aanpak in autonome navigatie is een sampling-gebaseerde Model Predictive Control (MPC) strategie, bekend als Model Predictive Path Integral control (MPPI). Dit algoritme benadert de optimale control sequence door te samplen uit een continue verdeling van inputs. Omdat het geen gradients gebruikt vereist MPPI alleen botsingscontrole om botsingen te vermijden en maakt het discontinuïteit in kostenfuncties mogelijk. Bovendien blijft de rekensnelheid grotendeels onaangetast door de complexiteit van de niet-lineaire dynamica van de robots, wat langere planningshorizons mogelijk maakt. Hoewel deze aanpak steeds populairder wordt in de motion planning onderzoek, blijft de toepassing ervan in dynamische omgevingen met meerdere interacting agents relatief onontgonnen.

Voortbouwend op de state-of-the-art in MPPI, introduceert Chapter 3 een Interaction-Aware Model Predictive Path Integral (IA-MPPI) controller die is afgestemd op drukke stedelijke grachten. Terwijl conventionele planners passief reageren op other agents, voorspelt en plant IA-MPPI actief en coöperatief in real-time, waarbij het 'freezing robot'-probleem wordt aangepakt en omgegaan wordt met discontinuïteit in navigatieregels. De gedecentraliseerde, communicatievrije architectuur gaat uit van samenwerking tussen agents en maakt gebruik van een tweedelige sample-evaluatie om de efficiëntie te verbeteren. Deze aanpak kan omgaan met niet-lineaire dynamica, arbitraire obstakel-geometrie

en langere planningshorizons, wat robuustheid en efficiëntie aantoont vergeleken met de state-of-the-art MPC in multi-agent omgevingen.

Hoewel in Chapter 3 een Constant Velocity model werd gebruikt voor het voorspellen van de doelen van other agents, introduceert Chapter 4 een learning-based trajectory prediction model dat is getraind op realistische kunstmatige data om de nauwkeurigheid in drukke omgevingen te verbeteren. Door de doelen van agents te extraheren uit voorspelde trajecten en deze informatie te integreren in de IA-MPPI controller, verbetert de prestatie van de planner aanzienlijk voor interacties in nauwe omgevingen. Bovendien presteert deze aanpak beter dan het beschouwen van het voorspelde traject als bezette ruimte, wat leidt tot veiligere navigatie in drukke stedelijke grachten.

Een van de belangrijkste nadelen van MPPI is dat het moeite heeft in zeer dynamische omgevingen omdat het gewoonlijk alleen sampelt rond een eerder plan. Wanneer er zich snelle veranderingen voordoen, zoals sterke verstoringen of een obstakel dat onverwacht de weg verspert, wordt het eerdere plan ongeldig en kunnen alle gegenereerde samples leiden tot botsingen. Om dit probleem aan te pakken, introduceert Chapter 5 Biased-MPPI, waarbij meerdere klassieke en learning-based aanvullende controllers worden geïntegreerd in de sampling-verdeling. Hoewel de wiskundige afleidingen aantonen dat dit een bias introduceert, verbetert het de reactiviteit, veiligheid en robuustheid ten opzichte van lokale minima aanzienlijk. Bovendien vermindert deze aanpak het aantal benodigde samples, waardoor de controller efficiënter wordt in termen van rekentijd.

Chapter 6 maakt gebruik van de gradientvrije aard van MPPI en past het toe op een ander domein dat contactrijke manipulation tasks omvat, die berucht moeilijk zijn voor model-based planning. Met een parallelle GPU-compatibele physics engine zoals IsaacGym kan MPPI efficiënt honderden inputreeksen parallel uitvoeren, sneller dan in real-time, om optimale controle te benaderen. Deze aanpak elimineert de noodzaak voor expliciete modellering van contactdynamica door gebruik te maken van de modellen die in de simulator zijn ingebouwd, waardoor het bijzonder geschikt is voor manipulation tasks zoals duwen en oppakken met zowel prehensiele als niet-prehensiele manipulators. Experimenten met echte robots tonen de effectiviteit van deze methode aan in het omgaan met discontinue, contactrijke omgevingen.

Al met al onderzoekt deze thesis belangrijke aspecten van bewegingsplanning, met name voor ASVs, met uitbreidingen naar grondrobots en mobiele manipulators. Het breidt het MPPI raamwerk uit voor autonome navigatie door het aan te passen voor multi-agent systemen en biases toe te voegen via klassieke en learning-based aanvullende controllers. Bovendien vergelijkt de thesis MPPI met MPC in navigatie-experimenten en met optimisation-fabrics in manipulation tasks. Het gepresenteerde werk bevordert het gebruik van MPPI, vooral in domeinen waar complexe niet-lineaire dynamica en discontinue kosten het gebruik van real-time optimisation-based MPC bemoeilijken.

Acknowledgments

Since childhood, I have always been curious. I have always wanted to understand things and figure out what I can do with them. Perhaps that should have been an early hint that a career in research and engineering was my path, but honestly, I wouldn't have figured it out without the people I met along the way. Now that this small chapter has ended, I have never been more excited about what will come. And if I now have a better idea of what I want, I owe it to all of you.

I want to thank my promotor and daily supervisor, Dr. Javier Alonso-Mora. I am very grateful for the opportunity you gave me, for trusting me, and for nudging me forward whenever I was stuck. Starting the PhD remotely during a pandemic was not ideal, but you were always available for discussion and ensured I had people to talk to when I needed to get my research started. I also want to thank my promotor, Prof. Robert Babuska, for the work behind the scenes and the interesting discussions over the past four years. I also appreciate Prof. Rudy R. Negenborn, Prof. Nathan van de Wouw, Prof. Cédric Pradalier, Dr. Alexander Liniger, and Prof. Martijn Wisse for taking the time to read my thesis and being part of my PhD committee.

My gratitude also goes to everyone involved in the TRiLOGy project. To Dr. Bilge Atasoy, Cigdem Karademir, and Dr. Breno Alves Beirigo, for the project organization, for the chats, and for being fantastic academic partners. To all the industry partners for proving their time, effort and funding in this endeavour. A special mention goes to Wouter Falkena and Fedor Ester from Demcon, Bart Verweijen from ZoevCity, Jonathan Klein Schiphorst from Roboat and Jochem Nonhebel from Damen Naval for helping with student projects and theses. I also want to mention Titus Venverloo from MIT Senseable City Amsterdam, Vittorio Garofano, and Bart Boogmans from ResearchLab Autonomous Shipping for their help setting up the boats and using their facilities.

To all the students I have supervised, Jitske, Jules, Tuhin, Lucas, Walter, Godert, Frankie, Hilte, and Jelmer: I have learnt more from you than you did from me. Working with you was a pleasure and good fun. You have all achieved great things already, but I'm sure even greater things will come. I am proud to have been a small part of your journey.

A thanks is due to all my co-authors and co-supervisors. To Bruno, for jump-starting my research. To Álvaro, for accepting my challenge and hour-long coffee chats while becoming an awesome friend in the process. To Corrado, for all the brainstorming, insights, and leads. To MaxS, for stubbornly showing everyone how things should be done. To Chadi, for always being the most valuable asset in the group. To Alex, for spending a month with me outside in the Dutch weather fixing the boats. To Xinwei, for giving me new directions, and to Khaled, for picking them up and pushing everyone to deliver. Learning from each other should be the ground rule of academia, and I have enjoyed every bit of that.

My ex-office mates, Anna, Luzia, Khaled, MaxiS, Saray, Nils, and Yujie, deserve a huge thanks for all the sweets, snacks, and teas we shared and all the dinners and activities we did together. Of course, tea breaks wouldn't have been the same without our honorary office mate Julian, ensuring we would never go too long without cake. I also want to thank my new office mates, Lorenzo, Sihao, Tasos, and Rodrigo, for putting up with me during the last phases of my PhD, and Lasse, for being a priceless source of tips and information. Most importantly, thank you all for the friendships we developed along the way.

The environment at Cognitive Robotics has been great, thanks to all of its people. I want to thank the other AMR members, Andreu, Dennis, MaxL, MaxiK, Ahmad, Daniel, Oscar, and Clarence, for our great discussions, dinners, and drinks. I want to thank Giovanni for his wisdom, Mariano for his energy, Jelle for his wittiness, and Manuel for humbling me (and Álvaro) at table tennis. I want to thank Tomás and Italo for being great people to have around and Domenico and Tomás for leaving a mark even if they were only here for a little while. I am also thankful to Alberto, Wilbert and Vishrut, who took me in despite sitting in a different wing of the building.

An acknowledgement is due to everyone else that made the last few years in Delft more enjoyable. Thanks to Patricjia, Jorge, and Fernanda for the hospitality and the good times. Thanks also to Natalia, Yuria, Tallo, BeaO, and BeaN (and Daniela!) for putting up with all the nerdy talk during the barbecues and dinners with a smile.

Now, without exaggeration, I really had the most awesome flatmates anyone could ever dream of during my time here in Delft. Corrado, since we randomly met in a hostel in Bologna and realized we were enrolled in the same class, you have been a guide and an inspiration. Daniel, the passion and commitment you put into everything you do is something I strive for and likely will never be able to match. Lucas, your cheerfulness and lightheartedness are second to none. Exploring Kiel's forests during our runs might be one of the main things that kept me sane during the pandemic. Hidde, I admire your integrity and open-mindedness. Our discussions at the dinner table were always something to look forward to, and I do miss them sometimes. Rubén, you are the kindest of all and a fabulous Monster Hunter companion. Sorry for bailing before the game finale! Hey, have I mentioned that we all decided to do a PhD? Talking about positive influence!

Un ringraziamento è dovuto anche Rosario, che nonostante la distanza degli ultimi anni, rimane sempre un punto fermo. Ce la faremo a salire sulla torre prima o poi Roh? Sarebbe poi impossibile non menzionare chi c'è stato sempre: Stefano e Mattia. Sono più di dieci anni che non viviamo vicini, eppure ogni volta che ci vediamo è come se fossimo sempre insieme. So di poter sempre contare su di voi, e per questo vi ringrazio. Eleonora, non so bene come fai a sopportare noi tre insieme. Grazie per essere sempre pronta ad accoglierci a braccia aperte con tutta la tua positività.

Dla Ani, Piotra, Claudii, Basi i Zdzisława. W ciągu ostatnich lat zawsze byliście gotowi przyjąć mnie z otwartymi ramionami do swojego domu i serca. Jestem za to bardzo wdzięczny i mam nadzieję na wiele kolejnych wspólnych lat.

A Franco, un ringraziamento per avermi sempre supportato e aver solleticato la mia curiosità sin da bambino. È sicuramente parte del motivo per cui sono arrivato a scrivere

questa tesi. A Gina, un grazie per l'energia, la solarità e gli aiuti in questi anni. A mio fratello, Matteo, un ringraziamento per essere stato sempre un'ispirazione e avermi insegnato ad affrontare i problemi usando la testa prima di chiedere aiuto. A mia mamma, Fiorella, il ringraziamento più grande per avermi sempre dato la possibilità di esprimermi, fare le mie scelte e supportarle nonostante tutte le difficoltà.

Finally, the most heartfelt thank you goes to my girlfriend, Daniela. I believe this is the fourth thesis I acknowledge you in, which must mean something. Many years, many countries, but you were always there when I needed you most, ready to give your love and support. I feel extremely lucky to have what we have. Kocham Cię!

Elia
Delft, March 2025

1

Introduction

This chapter positions the dissertation within the current societal context and its related challenges. Firstly, we identify our motivation in the pressing need to address the cost-effectiveness and operational efficiency of inland shipping, particularly in preserving canal-oriented urban centers such as Amsterdam. While Autonomous Surface Vessels (ASVs) could help, we highlight the importance of introducing interactions and regulation awareness in their motion planning. Lastly, we critically assess existing methodologies, delineate their limitations, and subsequently outline this dissertation's contributions and structure.

1

The city of Amsterdam was developed for transport over water, and many residential and business addresses have close access to its waterways. However, transport in inland waterways is often deemed more costly and labor-intensive than transport on wheels, and the canals remain underutilized. Meanwhile, the historic city center suffers from congestion on its road infrastructure. Heavy traffic, such as trucks for deliveries, construction, or garbage collection, stresses the city's delicate quay walls and foundation piles. Consequently, the city identified transport over water as one of the seven main activities and measures that can be used to improve urban transport and logistics [1].

Fully or partially Autonomous Surface Vessels (ASVs) could bring down the cost of transportation over inland waterways. Their use would relieve the city from the stress induced by excessive road traffic and allow for use cases that were not previously viable. For example, a fleet of ASVs could be used as garbage containers and autonomously drive themselves to the collection center once full [2].

Several companies are developing vessel autonomy. Depicted in Figure 1.1 are Roboat [3] and Zeabuz [4], which spun off from research projects [5, 6]. These two companies focus on autonomous navigation in urban canals to transport people, goods, or garbage collection. As of today, despite impressive demonstrations, the technology only allows for operation in relatively wide canals or fjords with few other agents, far from the congested canals of a city like Amsterdam.



(a) Roboat¹ for garbage collection.



(b) Zeabuz² for transport of people.

Figure 1.1: Examples of ASV currently under development in Amsterdam and Trondheim.

The main challenge lies in that, unlike roads, inland waterways are much less structured. The navigation rules are loosely defined, and vessels heavily rely on cooperation with other agents and interaction awareness to avoid collisions. Compared to a car, vessels' dynamics are more complex, thus requiring them to plan motion far into the future while quickly reacting to unexpected environmental changes.

Within autonomous navigation, these challenges fall onto the local motion planner. This component is responsible for navigating a vessel along a pre-determined path, but its key functions include fine-tuning the trajectory to ensure safe passage, avoiding collisions, and managing low-level control. Therefore, this dissertation focuses on local motion planning

¹Credits: MIT and AMS Institute

²Credits: Zeabuz

for ASVs in urban canals, addressing these unique challenges in dynamic, unstructured environments.

1.1 Limitations of Current Approaches

Several approaches have been proposed for ASVs. For example, Velocity Obstacles (VO) can be extended to include the rules defined by the COLLision avoidance REGulations at sea (COLREGs) [7]. However, such a method only reacts to the current state of the other agents in the environment, meaning that it cannot predict other agents' future motion and assumes all agents use the same planning method. Receding horizon planners are advantageous in environments with tight interactions. These planners can compute a plan and include predictions for the other agents within a certain time horizon.

At the beginning of our project, we explored the use of optimization-based Model Predictive Control (MPC) for ASVs in urban canals [8]. While applicable in some scenarios, this approach has several drawbacks. Firstly, the free space has to be decomposed into a convex region for the optimization to run in real-time, which can over-constrain the vessel's motion. Secondly, the navigation rules can only be introduced in the cost using smooth functions, which is more difficult than checking if a rule is broken. Thirdly, this planner receives a prediction of the other agents' future positions from a prediction module and then makes a plan for the ego agent that avoids those predicted states. This means that the planner is unaware that it can influence the behavior of other agents nor that other agents may cooperate in collision avoidance maneuvers.

Given a fixed set of motion primitives, it is possible to design a sampling-based MPC that accounts for COLREGs [9]. Given its gradient-free nature, this approach only requires collision checking to avoid collisions and allows for discontinuous cost functions. However, these motion primitives provide a too-coarse discretization of the input space to maneuver in narrow urban canals and are, therefore, only applicable in open waters.

Recently, another sampling-based MPC strategy, namely Model Predictive Path Integral control (MPPI) [10], has become popular in the robotics community. This algorithm approximates the optimal control sequence by sampling from a continuous distribution, but its application to dynamic environments with multiple agents remains underexplored.

1.2 Research Questions

We can narrow down the main objective of the dissertation to a single goal:

Main goal *To develop a local motion planner for ASVs capable of safe and efficient autonomous navigation in crowded urban canals.*

To tackle this problem, we need to break it down further. As discussed, the local motion planner must be interaction-aware, anticipating, and cooperating with other agents to resolve encounters. Additionally, it must incorporate navigation rules. Since these rules are often loosely defined in urban canals, including them in the cost function is more effective rather than as hard constraints.

1

However, defining these rules precisely can be difficult. It is often easier to identify when a rule has been broken (e.g., not respecting the right-hand rule in a head-on encounter or not giving the right-of-way in an intersection), which results in a sudden penalty in the cost function. This binary distinction between “rule respected” and “rule violated” leads to discontinuous costs.

Moreover, given the complex dynamics of vessels, the planner should be able to utilize a non-linear model of the system. Since vessels move slowly, the prediction horizon should be long enough to capture future states and anticipate necessary adjustments. Finally, the planner needs to run in real-time at a minimum frequency of 5-10Hz to ensure timely responses to environmental changes.

These requirements give rise to our first sub-question:

Question 1 *What is a suitable motion planner for interaction-rich urban canals?*

Moreover, data could be utilized by the planner to estimate some hidden parameters of the other non-communicating agents, such as their navigation goals. This leads to the second sub-question:

Question 2 *How can we leverage data to improve the performance of the planner?*

Lastly, while vessels move slowly, urban canals are very dynamic environments. The planner must react quickly to a swimmer jumping in the water or a vessel appearing out of an occluded corner. Moreover, considering interaction-rich scenarios, such a planner should also be able to swiftly switch out of local minima if the other agents don’t behave as expected, and another trajectory is then preferable. This observation leads to the third sub-question:

Question 3 *How can we improve efficiency and reactivity to unexpected changes in the environment?*

Tackling these research questions led to several scientific contributions, which will be discussed in the next Section.

1.3 Contributions

The overall contribution of this thesis is a motion planner for interaction-rich environments, which we apply to, among others, ASVs in urban canals. In particular, we extend MPPI to plan in multi-agent environments, leverage learning-based trajectory prediction algorithms, and use ancillary controllers to ensure safety and reactivity. These core contributions are detailed below.

Interaction-Aware MPPI We develop a decentralized and communication-free multi-agent MPPI controller capable of simultaneous prediction and planning under the assump-

tion of cooperation. The proposed planner satisfies the requirements used to formulate research question 1. It has a prediction horizon of 100 steps ahead at 10Hz planning with non-linear dynamics, discontinuous navigation rules, and interactions with other agents while accounting for the exact shape of static and dynamic obstacles.

Learning-Based Predictions We create an artificial dataset with ASVs navigating in urban canals over which we can adapt and train a trajectory prediction model. We then extract goals from the predicted trajectories and feed them to our Interaction-Aware MPPI, significantly improving its performances in dense, interaction-rich navigation experiments.

Biased-MPPI We provide mathematical derivations allowing for arbitrary changes in MPPI's sampling distribution while showing that doing so results in a bias in the solution. We leverage these novel derivations to introduce classical and learning-based ancillary controllers in the sampling distribution. This results in more efficient sampling, better robustness to environmental changes and local minima, and control fusion.

Isaac-MPPI We demonstrate how our approach applies to a different domain, where MPPI is used for contact-rich prehensile and non-prehensile manipulation. In this setting, we exploit a parallelizable physics engine, namely Isaac-Gym, to roll out hundreds of samples in parallel faster than in real time and use this experience to approximate the optimal control. The key advantage is that this approach requires minimal modeling effort, as contact dynamics are already encoded in the physics engine, and being gradient-free, the planner can easily deal with the discontinuous contact dynamics.

We also collaborated on developing an optimization-based MPC for urban canal navigation [8] and a task and motion planner for robot manipulation [11]. These results, however, are not included in this thesis. Together, our contribution advances the field of local motion planning, providing interesting applications, fundamental results, and open-source code that benefit the community.

1.4 Outline

Figure 1.2 depicts the outline of the dissertation. Chapter 2 lays out the foundational concepts of path integral control and its information-theoretic counterpart. Chapter 3 presents a decentralized multi-agent MPPI controller capable of simultaneous prediction and planning, addressing specific requirements for complex navigation scenarios. Chapter 4 extends the work in Chapter 3 by adapting and training a learning-based trajectory prediction model to provide better estimates of the other agents' hidden goals. Chapter 5 introduces the notion of biased sampling in MPPI, enabling improved sampling efficiency and robustness. Chapter 6 showcases the application of MPPI to contact-rich manipulation tasks using IsaacGym, a parallelizable physics engine, offering insights into efficient real-time control with minimal modeling. Lastly, Chapter 7 provides concluding remarks, summarizing the findings, discussing their implications, and suggesting avenues for future research.

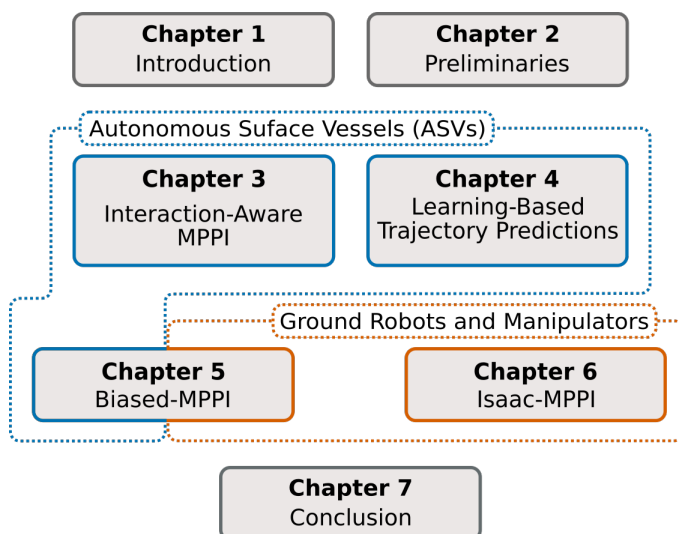


Figure 1.2: Structure of the document. Colors represent the type of robot on which the experiments were conducted.

2

2

Preliminaries

This chapter provides foundational background on the key concepts discussed in this dissertation. Its primary aim is to familiarize readers with Model Predictive Path Integral control (MPPI), covering its development from Stochastic Optimal Control (SOC) to its Information-Theoretic framework. We will briefly describe the main concepts, which are the groundwork for the contributions discussed in the following chapters. For a more in-depth description and discussion, the reader can refer to previous work [12].

2.1 Path Integral Control

Path integral control, first developed by Kappen [13, 14], is a method to solve nonlinear stochastic optimal control by estimating an expectation over system trajectories, e.g. through Monte Carlo sampling. Suppose to have dynamics in the form:

$$dx = [f(x_t, t) + G(x_t, t)u(x_t, t)]dt + B(x_t, t)dw \quad (2.1)$$

where $x_t = x(t) \in \mathbb{R}^N$ represents the state of the system at time t , $u(x_t, t) \in \mathbb{R}^m$ is the control input, and $dw \in \mathbb{R}^p$ is a Brownian disturbance. Note that the system model is affine in the control input. If we now denote $\phi(x_T, T)$ as a terminal cost, $q(x_t, t)$ as running cost, and $R(x_t, t)$ as a positive definite control cost matrix, the value function for this optimal control problem then is:

$$V(x_t, t) = \min_u \mathbb{E}_Q \left[\phi(x_T, T) + \int_t^T \left(q(x_t, t) + \frac{1}{2} u(x_t, t)^T R(x_t, t) u(x_t, t) \right) dt \right] \quad (2.2)$$

where $\mathbb{E}_Q[\cdot]$ is the expectation over controlled trajectories, i.e. with respect to equation (2.1). The Stochastic Hamilton-Jacobi-Bellman (HJB) equation then is:

$$-\partial_t V = \min_u \left[(f + Gu)^T \nabla_x V + \frac{1}{2} \text{tr}(BB^T \nabla_{xx} V) + q + \frac{1}{2} u^T R u \right] \quad (2.3)$$

with boundary condition $V(x_T, T) = \phi(x_T, T)$. The aforementioned minimization is now convex with respect to the control inputs u , and therefore the minimum can be found by taking the gradient with respect to u and equating it to zero, obtaining:

$$u^* = -R^{-1} G^T \nabla_x V. \quad (2.4)$$

If we plug this solution back into the Stochastic HJB equation, we obtain:

$$-\partial_t V = q + f^T \nabla_x V - \frac{1}{2} \nabla_x V^T G R^{-1} G^T \nabla_x V + \frac{1}{2} \text{tr}(BB^T \nabla_{xx} V). \quad (2.5)$$

If we could solve this equation for V , we could compute its gradient and find the optimal control input. Unfortunately, equation (2.5) is often a high-dimensional, non-linear partial differential equation, and standard techniques to solve it suffer from the curse of dimensionality and scale exponentially with the size of the state space. We can instead proceed to linearize equation (2.5) defining a desirability function $\psi(x, t)$ through:

$$V(x, t) = -\lambda \log(\psi(x, t)) \quad (2.6)$$

with λ a constant to be defined. After substitution and simplification, we obtain:

$$\partial_t \psi = \frac{\psi}{\lambda} q - f^T \psi_x - \frac{\lambda}{2\psi} \psi_x^T G R^{-1} G^T \psi_x - \frac{1}{2} \text{tr}(BB^T \psi_{xx}) + \frac{1}{2\psi} \psi_x^T BB^T \psi_x \quad (2.7)$$

where ψ_x is the derivative with respect to x . The terms quadratic in ψ disappear if and only if $BB^T = \lambda GR^{-1}G^T$. This restricts our choice of R , but this constraint results in a reasonable

control cost matrix that assigns a lower cost to those inputs that directly influence states that are affected by higher noise. If we can choose a suitable matrix R and constant λ , then the stochastic HJB equation is linearizable and it becomes:

$$\partial_t \psi = \frac{\psi}{\lambda} q - f^T \psi_x - \frac{1}{2} \text{tr}(BB^T \psi_{xx}). \quad (2.8)$$

The Feynman-Kac formula can now be applied to solve the Partial Differential Equation (PDE) as:

$$\begin{aligned} \psi(x_{t_0}, t) &= \mathbb{E}_{\mathbb{P}} \left[\exp\left(-\frac{1}{\lambda} S(\tau)\right) \right], \\ S(\tau) &= \phi(x_T) + \int_{t_0}^T q(x_t, t) dt \end{aligned} \quad (2.9)$$

where $S(\tau)$ is the state-dependent cost-to-go of the trajectory and $\mathbb{E}_{\mathbb{P}}[\cdot]$ is the expectation over the uncontrolled trajectories:

$$dx = f(x_t, t)dt + B(x_t, t)dw. \quad (2.10)$$

We can now express the optimal control with respect to ψ :

$$u^* = \lambda R^{-1} G^T \frac{\psi_x}{\psi}. \quad (2.11)$$

We now need to compute ψ_x . In the case that the system dynamics are neatly decomposed into indirectly and directly actuated components, i.e.:

$$G(x_t, t) = \begin{pmatrix} 0 \\ G_c(x_t, t) \end{pmatrix}, \quad B(x_t, t) = \begin{pmatrix} 0 \\ B_c(x_t, t) \end{pmatrix}, \quad (2.12)$$

such gradient can be found analytically [15] and the optimal control becomes:

$$u^* dt = R^{-1} G_c^T (G_c R^{-1} G_c^T)^{-1} B_c \frac{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\lambda} S(\tau) dw)]}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\lambda} S(\tau))]} \quad (2.13)$$

If the dynamics are not neatly decomposed, we can find a transformation matrix to achieve it. If the dynamics are not affine in control, we can make a control affine approximation by linearizing around a control sequence [12].

2.1.1 Monte Carlo Sampling

As described in equation (2.9), the expectation should be taken with respect to the uncontrolled dynamics. In Figure 2.1, we can see an example [13] of a particle that moves with constant horizontal velocity, while the vertical position is given by:

$$dx = udt + dw \quad (2.14)$$

with cost being quadratic around zero at final time 2 and infinite on the blue walls. The image shows the inefficiency of sampling from the uncontrolled dynamics, as almost all

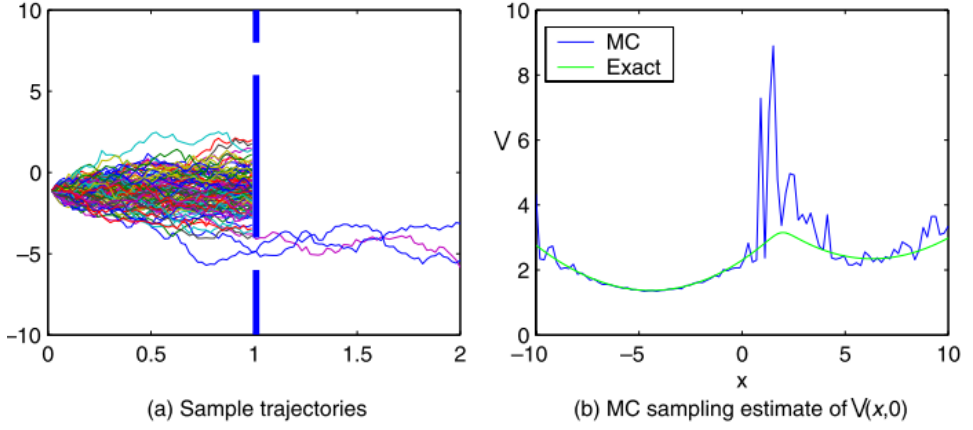


Figure 2.1: Monte Carlo sampling of $V(x, t=0)$ for the double-slit problem. (a) Sample trajectories starting at x to estimate $V(x, t)$. Most trajectories encounter an infinite cost. (b) MC estimate of $V(x, t=0)$ with $N = 100\,000$ trajectories for each x . Figure from [13] 6.1.1.

trajectories are killed by the infinite cost of hitting the wall, and only a few trajectories actually contribute to the estimate of the optimal value function.

Several algorithms to improve sampling have been proposed [15–19]. However, recent algorithms seem to focus on an Model Predictive Control (MPC)-like algorithm, MPPI, where at each time step, the algorithm samples around a control sequence given by the shifted optimal control computed at the previous time instant [20, 21].

2.2 Information-Theoretic Approach

Connections to free energy were noticed early on [13] as, substituting equation (2.9) into equation (2.6) we obtain:

$$V(x_{t_0}, t_0) = -\lambda \log \left\{ \mathbb{E}_{\mathbb{P}} \left[\exp \left(-\frac{1}{\lambda} S(\tau) \right) \right] \right\} = \mathcal{F}(S, \mathbb{P}, x_0, \lambda) \quad (2.15)$$

which is the definition of mechanical free energy with inverse temperature λ . These connections between path integrals and information theory were further developed [22, 23] and led to the design of information-theoretic MPC for SOC [24, 25]. To continue with this formulation, we need to define the relative entropy, also known as KL-Divergence. Given two probability distributions $\mathbb{P}$ and $\mathbb{Q}$, and assuming that the Radon-Nikodym derivative $\frac{d\mathbb{Q}}{d\mathbb{P}}$ exists, the relative entropy is defined as:

$$\text{KL}(\mathbb{Q} \parallel \mathbb{P}) = \mathbb{E}_{\mathbb{Q}} \left[\log \left(\frac{d\mathbb{Q}}{d\mathbb{P}} \right) \right]. \quad (2.16)$$

We can also express the expectation in the definition of the free energy with respect to a controlled distribution Q using the Radon-Nikodym derivative [26]:

$$\mathcal{F} = -\lambda \log \left(\mathbb{E}_Q \left[\exp \left(-\frac{1}{\lambda} S(\tau) \right) \frac{dP}{dQ} \right] \right) \leq -\lambda \mathbb{E}_Q \left[\log \left(\exp \left(-\frac{1}{\lambda} S(\tau) \right) \frac{dP}{dQ} \right) \right] = * \quad (2.17)$$

where we applied Jensen's inequality [27]. The rightmost term can be simplified as:

$$* = -\lambda \mathbb{E}_Q \left[-\frac{1}{\lambda} S(\tau) + \log \left(\frac{dP}{dQ} \right) \right] = \mathbb{E}_Q[S(\tau)] + \lambda \mathbb{E}_Q \left[\log \left(\frac{dQ}{dP} \right) \right] = \mathbb{E}_Q[S(\tau)] + \lambda \text{KL}(Q||P). \quad (2.18)$$

The following relationship between the free energy of a system and the relative entropy between a base measure P and a controlled measure Q has been found:

$$\mathcal{F}(S, P, x_0, \lambda) \leq \mathbb{E}_Q[S(\tau)] + \lambda \text{KL}(Q||P) \quad (2.19)$$

where $\mathbb{E}_Q[S(\tau)]$ is the expected state-dependent cost-to-go under the probability measure Q , and the KL-Divergence acts as control cost penalizing deviations of the controlled distribution Q from the uncontrolled distribution P . It can be proven that an optimal distribution Q^* in the sense that achieves the lower bound exists and has Radon-Nikodym derivative with respect to P given by:

$$\frac{dQ^*}{dP} = \frac{\exp(-\frac{1}{\lambda} S(\tau))}{\mathbb{E}_P[\exp(-\frac{1}{\lambda} S(\tau))]} \quad (2.20)$$

This means that, instead of solving the optimal control problem using the HJB equations, we can try to align our controlled distribution Q with the optimal distribution Q^* . If they are aligned, then sampling from Q will result in lower cost than any other control law. The existence of $\frac{dQ}{dP}$ is given by Girsanov's theorem, which allows us to compute the KL-Divergence as:

$$\text{KL}(Q||P) = \mathbb{E}_Q \left[\frac{1}{2} \int_0^T u_t^T G(x_t, t)^T \Sigma(x_t, t)^{-1} G(x_t, t) u_t dt \right] \quad (2.21)$$

where $\Sigma(x_t, t) = B(x_t, t)B(x_t, t)^T$. Combining this result with equation (2.19) we obtain:

$$\mathcal{F}(S, P, x_0, \lambda) \leq \mathbb{E}_Q \left[S(\tau) + \frac{\lambda}{2} \int_0^T u_t^T G(x_t, t)^T \Sigma(x_t, t)^{-1} G(x_t, t) u_t dt \right]. \quad (2.22)$$

Note that $R = \lambda G^T \Sigma^{-1} G$ was our choice of control cost matrix in path integral control to linearize equation (2.7). This means that in this information-theoretic framework, the control cost appears naturally and that the optimal distribution Q^* is optimal with respect to the standard cost used in path integral control. Therefore, the goal is just to find an input sequence U such that Q is as close as possible to the optimal distribution, and we can do this with:

$$U^* = \underset{U}{\operatorname{argmin}} \text{KL}(Q^*||Q). \quad (2.23)$$

While it is possible to find a solution for continuous-time systems that show connections to path integral control [12], it is more interesting to see how, in discrete time, we can use this framework for systems that are more general than the ones required by the classical path integral control. Consider the discrete-time dynamical system:

$$x_{t+1} = F(x_t, v_t) \quad (2.24)$$

where $x_t \in \mathbb{R}^n$ is the state vector and:

$$v_t \sim \mathcal{N}(u_t, \Sigma) \quad (2.25)$$

is the noisy input, with $u_t \in \mathbb{R}^m$ is the commanded input. Note that, by using this model, we do not assume affinity in control but we restrict ourselves to systems that only have noise directly inserted in the control input. Let's define $V = (v_0, v_1, \dots, v_{T-1})$ as a sequence of inputs over T time steps, and $U = (u_0, u_1, \dots, u_{T-1})$ as the sequence of commanded control inputs. We can now define the base measure $\mathbb{P}$ through its density function $p(V)$:

$$p(V) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m |\Sigma|)^{1/2}} \exp\left(-\frac{1}{2} v_t^T \Sigma^{-1} v_t\right). \quad (2.26)$$

Whereas density function for the controlled measure $\mathbb{Q}$ is expressed as:

$$q(V|U) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m |\Sigma|)^{1/2}} \exp\left(-\frac{1}{2} (v_t - u_t)^T \Sigma^{-1} (v_t - u_t)\right). \quad (2.27)$$

If we now have an initial condition x_0 and an input sequence V we can uniquely determine a trajectory τ by applying F recursively. For this we define a function $\mathcal{G}_{x_0} : \Omega_V \rightarrow \Omega_\tau$. If we define now a state-dependent cost function $C : \Omega_\tau \rightarrow \mathbb{R}^+$, we can find a cost function over input sequences $S : \Omega_V \rightarrow \mathbb{R}^+$ as the composition $S = C \circ \mathcal{G}_{x_0}$. The KL-Divergence between $\mathbb{Q}$ and $\mathbb{P}$ can be computed as:

$$\text{KL}(\mathbb{Q}||\mathbb{P}) = \mathbb{E}_{\mathbb{Q}} \left[\log \left(\frac{q(V|U)}{p(V)} \right) \right] = \frac{1}{2} \sum_{t=0}^{T-1} u_t^T \Sigma^{-1} u_t, \quad (2.28)$$

and the free-energy inequality becomes:

$$\mathcal{F}(S, \mathbb{P}, x_0, \lambda) \leq \mathbb{E}_{\mathbb{Q}}[S(V, x_0)] + \frac{\lambda}{2} \sum_{t=0}^{T-1} u_t^T \Sigma^{-1} u_t. \quad (2.29)$$

Meaning that, once again, the free-energy is a lower bound on the expected state dependent cost plus a quadratic control cost. As in the more general theoretical framework, we want to find an input sequence that minimizes the distance between $\mathbb{Q}$ and $\mathbb{Q}^*$. Using the

definition of KL-Divergence:

$$\begin{aligned}
 \text{KL}(\mathbb{Q}^* \parallel \mathbb{Q}) &= \int_{\Omega_V} q^*(V) \log \left(\frac{q^*(V)}{q(V|U)} \right) dV \\
 &= \int_{\Omega_V} q^*(V) \log \left(\frac{q^*(V)}{p(V)} \frac{p(V)}{q(V|U)} \right) dV \\
 &= \int_{\Omega_V} q^*(V) \log \left(\frac{q^*(V)}{p(V)} \right) - q^*(V) \log \left(\frac{q(V|U)}{p(V)} \right) dV,
 \end{aligned} \tag{2.30}$$

Where the first term in the integral does not depend on U . We can, therefore, write the optimization problem in equation (2.23) as:

$$\begin{aligned}
 U^* &= \operatorname{argmax}_U \int_{\Omega_V} q^*(V) \log \left(\frac{q(V|U)}{p(V)} \right) dV \\
 &= \operatorname{argmax}_U \int_{\Omega_V} q^*(V) \left(\sum_{t=0}^{T-1} -\frac{1}{2} u_t^T \Sigma^{-1} u_t + u_t^T \Sigma v_t \right) dV \\
 &= \operatorname{argmax}_U \sum_{t=0}^{T-1} \left(-\frac{1}{2} u_t^T \Sigma^{-1} u_t + u_t^T \int_{\Omega_V} q^*(V) \Sigma^{-1} v_t dV \right).
 \end{aligned} \tag{2.31}$$

Which is concave with respect to each u_t , so we can find the maximum with respect to each u_t by equating the gradient to zero. This yields to:

$$u_t^* = \int_{\Omega_V} q^*(V) v_t dV, \tag{2.32}$$

which reaches the intuitive conclusion that the optimal control that minimizes the KL-Divergence is the mean of the optimal distribution.

2.2.1 Iterative Importance Sampling

We can use importance sampling to obtain an unbiased estimate of the optimal control distribution given a current control distribution:

$$u_t^* = \mathbb{E}_{\mathbb{Q}^*}[v_t] = \int q^*(V) v_t dV = \int \frac{q^*(V)}{q(V|U)} q(V|U) v_t dV = \mathbb{E}_{\mathbb{Q}}[\omega(V) v_t], \tag{2.33}$$

with $\omega(V) = \frac{q^*(V)}{q(V|U)}$ being the importance sampling weight. This weight can be further split into:

$$\begin{aligned}
 \omega(V) &= \left(\frac{q^*(V)}{p(V)} \right) \left(\frac{p(V)}{q(V|U)} \right) \\
 &= \frac{1}{\eta} \exp \left(-\frac{1}{\lambda} S(V) \right) \left(\frac{p(V)}{q(V|U)} \right),
 \end{aligned} \tag{2.34}$$

with $\eta = \int \exp\left(-\frac{1}{\lambda}S(V)\right)p(V)dV$. By taking the distributions as in equation (2.26) and equation (2.27) we have:

$$\begin{aligned} \frac{p(V)}{q(V|U)} &= \frac{\exp\left(-\frac{1}{2}\sum_{t=0}^{T_1} v_t^T \Sigma^{-1} v_t\right)}{\exp\left(-\frac{1}{2}\sum_{t=0}^{T-1} (v_t - u_t)^T \Sigma^{-1} (v_t - u_t)\right)} \\ &= \exp\left(-\frac{1}{2}\sum_{t=0}^{T-1} u_t^T \Sigma^{-1} u_t + 2\epsilon_t^T \Sigma^{-1} u_t\right), \end{aligned} \quad (2.35)$$

where we have rewritten $v_t = u_t + \epsilon_t$. We can finally write down the update rule:

$$\begin{aligned} \omega(V) &= \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}\left(S(V) + \frac{\lambda}{2}\sum_{t=0}^{T-1} u_t^T \Sigma^{-1} u_t + 2u_t^T \Sigma^{-1} \epsilon_t\right)\right) \\ u'_t &= \mathbb{E}_Q[\omega(V)v_t] \\ u^* &= u'_0 \end{aligned} \quad (2.36)$$

We have derived an update rule for the optimal control estimation within the MPPIcontrol framework, using importance sampling to approximate the optimal control distribution. The weight $\omega(V)$ encapsulates each sampled trajectory's contribution, which is influenced by both the state-dependent cost $S(V)$ and a quadratic control cost. This relationship ensures that trajectories that minimize the state and control are given higher importance in determining the optimal action.

The update rule we derived shows how the control sequence is adjusted iteratively, considering the stochastic perturbations ϵ_t and the feedback from the predicted cost. This formulation allows us to efficiently explore the control space by sampling around the nominal trajectory and iteratively improving the control sequence. Ultimately, this method provides an efficient way to compute optimal control actions for any system, including discontinuities and non-linearities in the dynamics or cost function.

3

3

Interaction-Aware Local Motion Planning

Urban canals are narrow and crowded, necessitating adherence to navigation rules by vessels. Model Predictive Control (MPC) is a promising approach as it can incorporate predictions of other agents' motions and integrate navigation rules. However, real-time implementation of MPC requires several approximations. Typically, an external prediction algorithm provides trajectories for other agents, which the ego agent uses to plan its own trajectory, unaware of its influence on them. Additionally, static obstacle maps must be simplified into convex free spaces, and navigation rules must be transformed into linear constraints or continuous cost regions. These approximations can result in potentially hazardous situations and deadlocks.

This chapter introduces a decentralized, communication-free, Interaction-Aware Model Predictive Path Integral (IA-MPPI) controller that performs simultaneous prediction and planning while considering discontinuous navigation rules. Section 3.1 provides further motivation for this work, with a brief review of state-of-the-art approaches and a statement of our contributions. Section 3.2 presents the notation for Model Predictive Path Integral control (MPPI), the modeling of the vessel, and the global planning algorithm. Our approach and the core components that enable its functionality are detailed in Section 3.3. Experiments and comparisons with optimization-based MPC are discussed in Section 3.4, focusing on performance and computational complexity. Finally, Section 3.5 concludes the chapter.

This chapter is a verbatim copy of the peer-reviewed paper [28]:

-  L. Streichenberg*, E. Trevisan*, J. J. Chung, R. Siegwart and J. Alonso-Mora, "Multi-Agent Path Integral Control for Interaction-Aware Motion Planning in Urban Canals," 2023 IEEE International Conference on Robotics and Automation (ICRA). *Indicates equal contribution in alphabetical order.

Statement of contributions: Elia contributed to the initial idea of decentralizing MPPI and extracting goals from a constant velocity model and provided theoretical insights. Under Elia's supervision, Lucas formulated the two-stage sampling strategy and led the software development. Elia and Lucas contributed equally to the experiments and the writing of the final manuscripts. Jen Jen, Roland, and Javier provided discussions on the ideas and feedback, as well as editing of the manuscript.

3.1 Introduction

With rising population density, cities are forced to enhance their mobility and transportation strategies. The City of Amsterdam aims to reduce the load on road infrastructure by transporting goods and people on the urban waterways [29]. This presents a great opportunity to operate Autonomous Surface Vessels (ASVs) such as Roboat [30] in urban canals. However, this is a very technically challenging task due to the complex and dynamic nature of the environment. Narrow canals, complex dynamics, static obstacles, and human-piloted vessels must be dealt with while obeying existing canal regulations [31]. MPPI [20] offers a parallelizable sampling-based framework for solving motion planning tasks with such complex dynamics and discontinuous costs as those exhibited in our domain. Unlike methods based on constrained optimization, which need to rely on convex approximations of the free space and on inflating the ego and obstacle agents into ellipsoidal shapes for collision avoidance [8], MPPI can account for the exact and potentially non-convex shape of both static and dynamic obstacles. This promises to be a significant advantage in tight interaction-rich environments. Still, another important aspect of achieving safe and efficient navigation in crowded spaces is accounting for cooperation [32]. For these reasons, we propose a method to decentralize MPPI in order to navigate among non-communicating agents while providing interaction awareness and generating cooperative motion plans. We introduce awareness of navigation rules through discontinuous costs. Moreover, the proposed method can run in real-time thanks to our two-stage sample evaluation strategy and CPU parallelization.

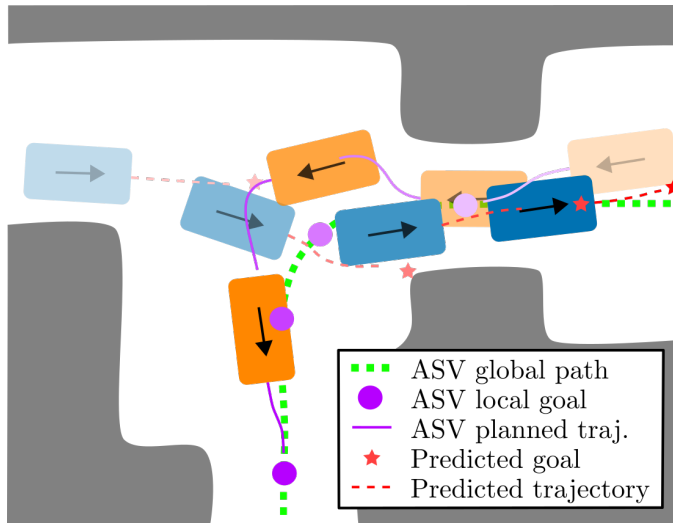


Figure 3.1: An ASV running our method (orange) encounters a non-communicating vessel (blue). At each time step, the ASV is given its current position and a global path (dashed green line). Based on this, the ASV sets a local goal (purple circle) in front of itself on the global path. The ASV is also given the position and velocity of the non-communicating vessel. With these, the ASV estimates the local goal of the non-communicating vessel (red star) assuming constant velocity. Then, the ASV plans input sequences over a defined horizon resulting in trajectories for both vessels.

3.1.1 Related Work

Cooperative and interactive motion planning for robotics is a challenging problem with a vast literature [33].

The most well-known examples for planning in dynamic environments are the Dynamic Window Approach (DWA) [34], Reciprocal Velocity Obstacles (RVO) [35], its extension Optimal Reciprocal Collision Avoidance (ORCA) [36], and Artificial Potential Fields (APF) [37, 38]. While these methods are highly efficient, they often lead to reactive behaviors.

Model-free reinforcement learning algorithms have been successfully trained in simulation with hand-crafted reward functions to navigate among human crowds [39, 40], but generalization and collision avoidance are not guaranteed.

Game theoretic approaches have been implemented in the context of autonomous cars to perform lane changes [41], merging [42] and to solve unsignalized intersections [43]. However, they rely on a coarse discretization of the action space and do not scale well with the number of agents.

MPC based on trajectory optimization is a popular approach when it comes to local motion planning. In a multi-agent setting, however, the actions of the other agents are required to proceed with the motion planning of the ego-agent. In [44] a distributed MPC was developed for motion planning with multiple drones relying on ideal communication. A way to avoid communication is to estimate each agent's state and predict their future motion using, for example, a constant velocity model [8, 45] or learning-based techniques [46]. However, as long as we first predict and then plan, large parts of the state-space can be perceived as unsafe by the motion planner [47]. This can be overcome by modeling interaction, such that the ego-agent is aware that its actions can influence the actions of the other agents around it. Unfortunately, accounting for such a model while planning with constrained optimization techniques can become computationally expensive [48].

In contrast to optimization-based methods, MPPI solves for the best control trajectory at each step by forward simulating the behavior of the full system. To achieve this, MPPI uses a parallelizable sampling-based framework [13] to rollout simulations, allowing it to find an approximate solution to non-linear, non-convex, discontinuous Stochastic Optimal Control (SOC) problems [10, 20]. Compared to other SOC methods such as iterative Linear Quadratic Gaussian (iLQG) [49] or Differential Dynamic Programming (DDP) [50], MPPI does not require linearization of the system dynamics or quadratic approximation of the cost function. This makes MPPI particularly well-suited to our target task of ASV navigation in urban canals since the regulation-based interactions explicitly give rise to non-differentiable costs. Moreover, MPPI's fast parallelizable computations could solve interaction-aware motion planning problems while still running in real-time. When deployed to centralized multi-agent systems, however, the classic MPPI approach shows a significant increase in the number of samples required with an increasing number of agents [51].

3.1.2 Contributions

We propose an interaction-aware motion planning method based on MPPI which can generate cooperative plans in environments with non-communicating agents accounting for

the full dynamics of the system and the exact shapes of the obstacles. To summarize our contributions:

- We propose a decentralized architecture that can operate with limited or no communication under the assumption that other agents' states can be sensed exactly and that all the agents in the environment behave rationally.
- To reduce the number of samples required to plan for multi-agent systems, we propose a two-stage sample evaluation technique that improves sample efficiency.
- We formulate the objectives of the navigation task into an appropriate cost function to achieve rule compliance.

To demonstrate the performance of our method, we compare it to a state-of-the-art regulations-aware optimization-based MPC [8] in several simulated experiments set up in real sections of Amsterdam's canals. The proposed decentralized MPPI is also compared to the centralized version in environments with crowds of up to four interacting vessels. Finally, we demonstrate the robustness of the algorithm in scenarios where a human-driven vessel does not behave rationally and provide insights into the computation times.

3.2 Preliminaries

We start by describing the basic MPPI algorithm. Then, since MPPI relies on a model of the system to perform the simulated rollouts, we also define the relevant dynamics that describe the behavior of our multi-ASV system.

3.2.1 MPPI Algorithm

The presented work is based on the MPPI derivations by [10]. With this method, we can solve SOC problems for discrete-time dynamical systems of the form,

$$\mathbf{q}_{t+1} = \mathcal{F}(\mathbf{q}_t, \tilde{\mathbf{u}}_t), \quad \tilde{\mathbf{u}}_t \sim \mathcal{N}(\mathbf{u}_t, \Sigma), \quad (3.1)$$

with state $\mathbf{q}$, time step t , nonlinear state transition function $\mathcal{F}$ and noisy input $\tilde{\mathbf{u}}$ with variance Σ and mean $\mathbf{u}$, where $\mathbf{u}$ is the input we command to the system. The algorithm samples K input sequences $\tilde{\mathbf{U}}_k$, $k \in [1, K]$ from a distribution $\mathcal{N}(\mathbf{u}_t, \nu\Sigma)$ (with scaling parameter ν) [12] and simulates them into K state trajectories Q_k over a horizon T as,

$$Q_k = [\mathbf{q}_0, \mathcal{F}(\mathbf{q}_0, \tilde{\mathbf{u}}_{k,0}), \dots, \mathcal{F}(\mathbf{q}_{k,T-1}, \tilde{\mathbf{u}}_{k,T-1})]. \quad (3.2)$$

Each sample is rated by computing the total cost S_k , which includes a stage cost and a terminal cost. *Importance sampling* weights w_k are then computed based on the cost of the sample k minus the minimum sampled cost S_{min} as,

$$w_k = \frac{1}{\eta} \exp\left(\frac{-1}{\lambda}(S_k - S_{min})\right), \quad \sum_{k=0}^{K-1} w_k = 1, \quad (3.3)$$

with normalization factor η and tuning parameter λ . The resulting control input sequence U^* , which approximates the optimal control input sequence, is computed with,

$$U^* = \sum_{k=0}^{K-1} w_k \tilde{U}_k. \quad (3.4)$$

Then, the first input u_0^* of the computed sequence U^* is applied to the system.

3.2.2 Vessel State and Dynamics

We define the multi-agent state similarly to [51]. The state of agent i is defined as the concatenation of its position, heading, and associated velocities,

$$\mathbf{q}_i = [\mathbf{x}_i^\top \quad \mathbf{v}_i^\top]^\top = [x_i \quad y_i \quad \phi_i \quad \dot{x}_i \quad \dot{y}_i \quad \dot{\phi}_i]^\top. \quad (3.5)$$

The full system state is then formed by stacking the individual states of each of the agents in the set $\mathcal{M} = \{0, 1, \dots, m\}$,

$$\mathbf{q} = [\mathbf{q}_0^\top \quad \mathbf{q}_1^\top \quad \dots \quad \mathbf{q}_m^\top]^\top. \quad (3.6)$$

The ASV we use is modeled as a nonlinear second order system [52–54]. Since the vessel sails at low speeds, we discard Coriolis and centripetal effects. Therefore,

$$\begin{aligned} \dot{\mathbf{x}}_i &= \mathbf{R}(\phi_i) \mathbf{v}_i, \\ \dot{\mathbf{v}}_i &= \mathbf{M}^{-1} (\mathbf{B} \mathbf{u}_i - \mathbf{D}(\mathbf{v}_i) \mathbf{v}_i), \end{aligned} \quad (3.7)$$

where $\mathbf{R}(\phi_i)$ is the rotation matrix from body to inertial frame, $\mathbf{M}$ is the mass matrix, $\mathbf{B} \mathbf{u}_i$ are the applied forces (thrust) and $\mathbf{D}(\mathbf{v}_i)$ is the drag matrix.

3.2.3 Global Planning

To help navigate large maps, we provide the ego-agent with a global path. Such a path is generated via the Robot Operating System (ROS) navigation stack with its path planning plugin [55]. Instead of directly tracking this global path, we look for a local goal $\mathbf{p}_g$ on the global path at a given look-ahead distance $r_{\mathbf{p}_g}$ (Fig. 3.5). Compared to just rigorously tracking the global path, this local goal approach gives the local planner more freedom to perform collision avoidance and other maneuvers.

3.3 Decentralized Interaction-Aware MPPI

In the following we outline the proposed architecture, state the changes to the classic MPPI framework and present the regulation-aware cost function along with the local goal prediction used for the decentralized computation of the cost.

3.3.1 Approach and Architecture

Our decentralized MPPI approach relies on each agent running its own MPPI solver for its local multi-agent system to anticipate the actions of other agents (see Fig. 3.2). That is,

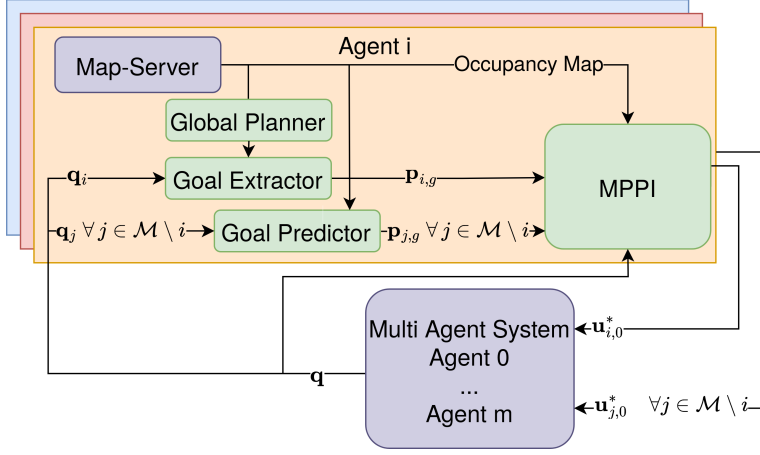
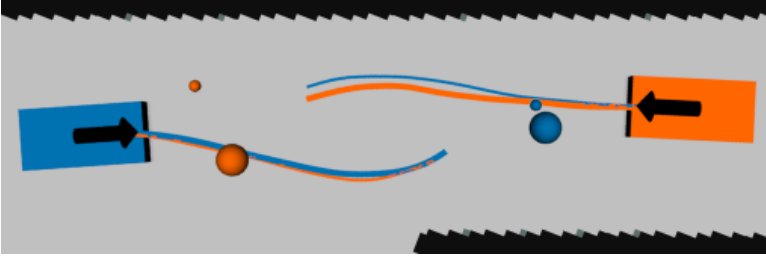


Figure 3.2: Overview of the framework: At every time step, each agent receives the state of all the agents in the environment. With this, we extract a local goal for the ego-agent (agent i in figure) from the global planner and predict local goals for the other agents. We then solve the planning task with MPPI as if it was centralized, planning a control input sequence and corresponding trajectory for each agent. We then apply the first input $u_{i,0}^*$ of the sequence to the ego-agent.

for agent i , the MPPI state and control output are defined as,

$$\begin{aligned} \mathbf{q}^i &= \begin{bmatrix} \mathbf{q}_i^\top & \mathbf{q}_j^{i\top} \end{bmatrix}^\top, \\ \mathbf{u}^i &= \begin{bmatrix} \mathbf{u}_i^\top & \mathbf{u}_j^{i\top} \end{bmatrix}^\top, \end{aligned} \quad \forall j \in \mathcal{M} \setminus i, \quad (3.8)$$

where $(\cdot)_j^i$ signifies a variable that agent i estimates of agent j . In the centralized case described in Section 3.2.1, the system state is fully observable and the inputs computed by the central controller will be those executed by each agent. When we move to the decentralized case, the positions and velocities of other agents must be communicated or observed. Furthermore, while each agent samples control actions for all other agents in the MPPI rollouts, at execution time, there is no guarantee that other agents will behave accordingly. To focus on the decentralized coordination problem, we make a few assumptions. First, we assume noise-free observations of the positions and velocities of other agents, i.e. $\mathbf{q}_j^i = \mathbf{q}_j$. Second, we assume that all agents behave rationally and that they are minimizing the same global cost. We later show in our experiments that the controller is still able to perform well when this assumption is violated. Third, in our experiments we only consider scenarios with homogeneous agents, meaning that they all have the same dynamics. This third assumption is not required in general as considering different dynamical models for different agents is possible as long as models are known. Fig. 3.3 shows a simulated encounter between two ASVs running our decentralized algorithm without communication.



3

Figure 3.3: Screenshot of two agents running our decentralized method with no communication inside our simulator. The left (blue) agent is given a local goal (large blue ball), its state, and the state of the other (orange) agent, based on which it predicts a local goal (small orange ball) for the obstacle vessel. The blue agent plans a sequence of inputs for both itself and the obstacle agent resulting in two trajectories, depicted by the blue paths. In orange, we can see the other agent applying the same algorithm. Note that even though we use constant velocity to predict the goal of the obstacles, both agents plan cooperation in the collision avoidance.

3.3.2 Two-stage Sample Evaluation

With an increasing number of agents, it is increasingly likely for at least one agent to collide with a static obstacle in most rollouts. In the classical implementation of MPPI, this leads to most rollouts receiving a high cost and being effectively rejected, resulting in a very low sample efficiency. Therefore we propose to decouple the sampling into two stages as shown in Algorithm 1. Control-samples $U_{j,k}$ are evaluated in parallel for every agent $j \in \mathcal{M}$, predicting the set of individual trajectories $Q_{j,k}$ with agent-centric costs $S_{j,k}$ (eq. 3.9). At this point all samples with cost larger than the collision penalty $C_{\text{collision}}$ are immediately discarded (lines 3 and 4 of Algorithm 1). To build the expected number of system samples K we sample uniformly from the remaining non-colliding samples of each agent and unify these into full system samples, i.e. by stacking as in eq. (3.6). For each system sample Q_k the complete configuration cost S_k is evaluated by adding the stored agent-centric costs $S_{j,k}$ with any costs arising from collisions between vessels and regulation violations (line 14, Algorithm 1).

3.3.3 Cost Formulation

The sample cost $S_k, \forall k \in [1, K]$ evaluation is split into agent-centric and configuration costs. Both are considered instantaneous costs and are evaluated for every time-step t within the horizon T and each sample k . The **agent-centric** cost $S_{j,k,t}$ (in the following S_{agent}) is evaluated for agent j for sample k and defined as,

$$S_{\text{agent}} = C_{\text{static}} + C_{\text{rotation}} + C_{\text{tracking}} + C_{\text{speed}} + C_{\text{sample}}. \quad (3.9)$$

C_{static} returns a constant penalty $C_{\text{collision}}$ if the vessel enters occupied space and C_{rotation} is based on a linear penalty for rotation velocities ($k_{\text{rot, slow}}$ for velocities $\|\mathbf{v}\|_2 < 0.5\text{m/s}$, k_{rot} otherwise). The tracking cost is,

$$C_{\text{tracking}} = k_{\text{tracking}} \frac{\|\mathbf{p}_g - \mathbf{p}_t\|_2}{\|\mathbf{p}_g - \mathbf{p}_{t_0}\|_2}, \quad (3.10)$$

Algorithm 1 Decentralized MPPI for agent i (agent-specific superscripts are dropped for clarity)

Require: U ▷ previous control sequence (hot start)
Require: q ▷ current system state

```

1:  $\mathbf{p}_{i,g} \leftarrow \text{receiveEgoLocalGoal}()$ 
2:  $\mathbf{p}_{j,g} \leftarrow \text{predictLocalGoal}(\mathbf{q}_j)$  ▷  $\forall j \in \mathcal{M} \setminus i$ 
3: for each agent  $j \in \mathcal{M}$  do ▷ independent rollouts
4:   for each sample  $k$  do
5:      $\mathcal{E}_{j,k} = [\epsilon_0, \dots, \epsilon_{T-1}]$  ▷  $\epsilon_t \in \mathcal{N}(0, \nu\Sigma)$ 
6:      $\tilde{U}_{j,k} = U_j + \mathcal{E}_{j,k}$ 
7:      $Q_{j,k} \leftarrow \text{simulateSystem}(\mathbf{q}_j, \tilde{U}_{j,k})$ 
8:      $S_{j,k} \leftarrow \text{getIndividualCost}(Q_{j,k}, \tilde{U}_{j,k}, \mathbf{p}_{j,g})$ 
9:     if  $S_{j,k} > C_{\text{collision}}$  then
10:        $\text{discardSample}(Q_{j,k}, \tilde{U}_{j,k}, \mathcal{E}_{j,k}, S_{j,k})$ 
11:     end if
12:   end for
13: end for
14: Uniformly sample from the remaining valid input sequences to rebuild  $K$  full system
    samples
15: for  $k = 1 : K$  do
16:    $\mathcal{S}_k = \sum_{j \in \mathcal{M}} S_{j,k} + \text{getConfigurationCost}(Q_k)$ 
17: end for
18:  $[w_1, \dots, w_K] = \text{importanceSampling}([\mathcal{S}_0, \dots, \mathcal{S}_K])$ 
19: return  $U^* = U + \sum_{k=1}^K w_k \mathcal{E}_k$ 

```

where k_{tracking} is a scaling factor, $\mathbf{p}_g$ is the agent's local goal, $\mathbf{p}_t$ is the predicted agent position at time t and $\mathbf{p}_{t_0}$ is the vessel position at the start of the prediction horizon. C_{speed} is a constant penalty applied when the current speed is higher than the maximum speed. The sample cost is given by,

$$C_{\text{sample}}(\mathbf{u}_t, \epsilon_t) = \frac{1}{2} \gamma [\mathbf{u}_t^T \Sigma^{-1} \mathbf{u}_t + 2\mathbf{u}_t^T \Sigma^{-1} \epsilon_t], \quad (3.11)$$

with γ as a tuning parameter. The **configuration cost** $S_{k,t}$ (in the following $S_{\text{configuration}}$) is evaluated for every timestep t within the horizon T for every sample k and combines dynamic collisions and regulation violations,

$$S_{\text{configuration}} = C_{\text{dynamic}} + C_{\text{regulation}}. \quad (3.12)$$

Dynamic collisions are defined as those between multiple vessels and are penalized with the same constant $C_{\text{collision}}$ and the regulation cost $C_{\text{regulation}}$ is derived from the two main traffic rules (i) *avoiding to the right* in head-on encounters and (ii) *right of way* for crossing scenarios similar to COLREGs [56]. Regulation compliance is determined by a relative position and relative velocity check. Regarding the position, we check if there is

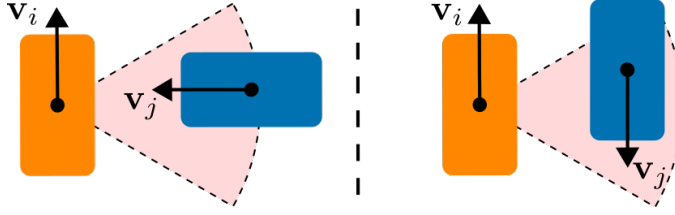


Figure 3.4: Configurations considered as regulation violations. Left: Not giving right of way to a vessel approaching from starboard. Right: Avoiding an oncoming vessel to the left.

3

a vessel with significant velocity ($\|\mathbf{v}\| > 0.5m/s$) on starboard side within a given radius (Fig 3.4). Regarding the relative velocities, we evaluate if another vessel approaches from the right by,

$$\|\mathbf{v}_i \times \mathbf{v}_j\| < \|\mathbf{v}_i\| \|\mathbf{v}_j\| \sin(-\frac{\pi}{2} + \delta), \quad (3.13)$$

where δ defines the angular margin, and if the other vessel is approaching the ego-vessel head-on via,

$$\|\mathbf{v}_i \cdot \mathbf{v}_j\| < \|\mathbf{v}_i\| \cdot \|\mathbf{v}_j\| \cos(\pi + \delta). \quad (3.14)$$

Therefore if we detect a vessel on starboard side and the velocities satisfy (3.13), we consider the ego-agent as breaking the right-of-way rule (Fig. 3.4 left). If instead, we detect a vessel on starboard side with opposite velocity, we consider it as passing on the right (Fig. 3.4 right).

3.3.4 Local Goal Prediction

The proposed decentralized version of interaction-aware MPPI requires estimating the local goals for all non-ego vessels (line 2, Algorithm 1). We use a constant velocity model such that agent i estimates the goal of agent j as,

$$\mathbf{p}_{j,g}^i = k_s T \delta T \mathbf{R}(\phi_j^i) \mathbf{v}_j^i + \mathbf{p}_j^i, \quad (3.15)$$

with k_s as a scaling factor and δT as step size. If the predicted goal is in collision, we project back along the vector towards $\mathbf{p}_j^i$ and choose the first unoccupied point as shown in Fig. 3.5.

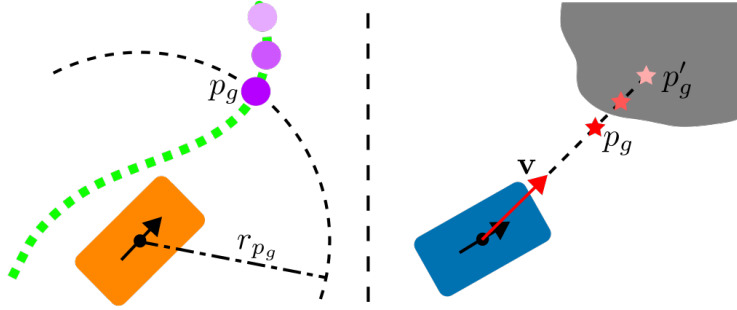


Figure 3.5: Left: Extraction of the local goal of the ego vessel is performed by searching the global path backward until the goal position is within a radius r_{p_g} from the center of the vessel. Right: Local goal prediction for other agents is performed using a constant velocity model, then projected into unoccupied space if the goal is in collision with static obstacles.

3.4 Experiments

We perform extensive experiments in several maps taken from real canal sections of Amsterdam, namely the Herengracht (HG), Prinsengracht (PG), Bloemgracht (BG), and the intersection between Bloemgracht and Lijnbaansgracht (BGLG). In all experiments, the dimensions of both the map and the vessel are represented faithfully. In Section 3.4.1 we compare our method in two-agent scenarios with an optimization-based MPC, in Section 3.4.2 we test our method in interaction-rich four-agent scenarios, in Section 3.4.3 we demonstrate the robustness to non-rational human-driven agents, while in Section 3.4.4 we discuss the computation times of the proposed method. All experiments running MPPI use a horizon T of 100 time-steps with step-size $\delta T = 0.1s$, input variance $\Sigma = \text{diag}(0.5, 0.5, 0.01, 0.01)$ and exploration scaling factor $\nu = 12$. We use $K = \{2000, 6000\}$ samples for two- and four-agent scenarios, respectively. Each version of the MPPI shown in the experiments (centralized, decentralized, decentralized with no communications) uses our proposed two-stage sampling technique.

3.4.1 Comparison with Optimization-based MPC

We compare our method to a state-of-the-art optimization-based decentralized motion planner designed for ASVs in urban canals, namely the Regulations Aware Model Predictive Contouring Controller (RA-MPCC) [8]. The three scenarios on which we compare are an unprotected left turn (Fig. 3.6a), a head-on encounter (Fig. 3.6b) and a crossing (Fig. 3.6c). These three scenarios were then run 100 times with randomized initial conditions and global goals using the proposed centralized, decentralized, and decentralized with no communication MPPI as well as the RA-MPCC. For all controller types, the randomization was kept equal (i.e. same random seed). Table 3.1 summarizes the results, where we compare the number of runs that ended successfully, in a deadlock, or in a collision. Of the runs that ended successfully, we report the number of rule violations. We also report the average time to complete the scenario, defined as the moment in which all agents reach their goal, and the total average distance, which is the sum of the average

distances traveled by all agents.

All controllers were encouraged through the cost function to keep a velocity around the speed limit (around 1.7m/s). From the table, however, it stands out that the RA-MPCC navigates much slower and therefore has much longer arrival times. This is both because of how its cost function is defined, but also because the RA-MPCC has to plan within convex obstacle-free areas, which can sometimes be quite small and slow down the pace. Instead, MPPI considers the exact occupancy map without any need for pre-processing.

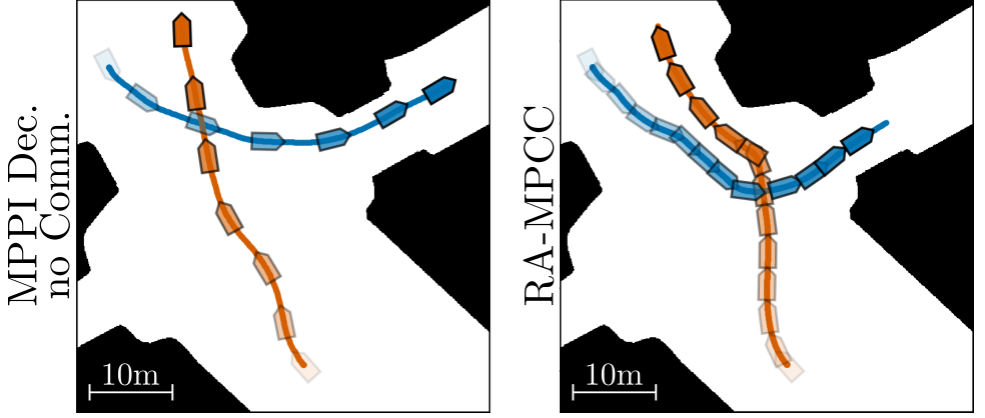
While it is easy to give a discontinuous penalty to the MPPI whenever a sample violates a navigation rule, the RA-MPCC has to use continuous cost functions to encourage rule compliance. This, however, inadvertently introduces repulsive forces between the two agents, which then tend to push each other into corners, which is the main cause of deadlocks in the Prinsengracht and the Bloemgracht-Lijnbaansgracht scenarios. In the Bloemgracht head-on encounter, the RA-MPCC gets to its destination only about half of the time. Given that the RA-MPCC has to inflate the ego-agent in a set of circles, the obstacle agent into an ellipsoid, and the static obstacle map has to be pre-processed into convex regions, there is barely enough space to pass in the most narrow section of the canal. This, combined with the lack of understanding that the two agents can cooperate to solve the maneuver, leads to a large number of deadlocks.

Collisions with the RA-MPCC instead happen for two reasons. Number one, the method first approximates the static obstacle with polygons, which are then decomposed into convex shapes, to which we can then find linear constraints by solving a quadratic program. However, there is no guarantee that the polygons contain all of the original obstacle. Safety margins are added, but margins too large means that some narrow canals are simply impossible to navigate. Number two, the optimization can often just fail, especially in more difficult and risky situations. When this happens, the algorithm just applies zero input, and if the boat has enough momentum it can drift into a collision.

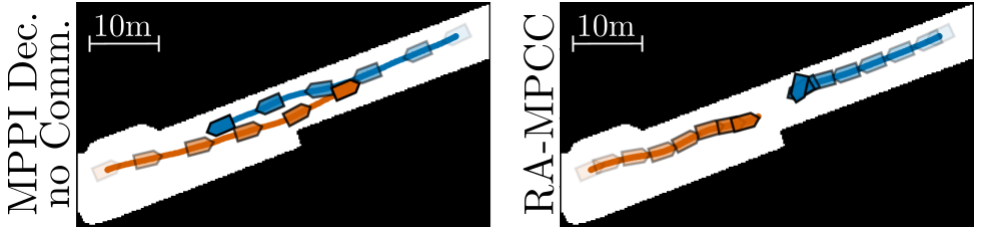
Moreover, while our interaction-aware MPPI could plan with a horizon of 100 steps, the RA-MPPC could only plan 20 steps to meet the 10Hz control loop.

3.4.2 Navigation in Crowded Environments

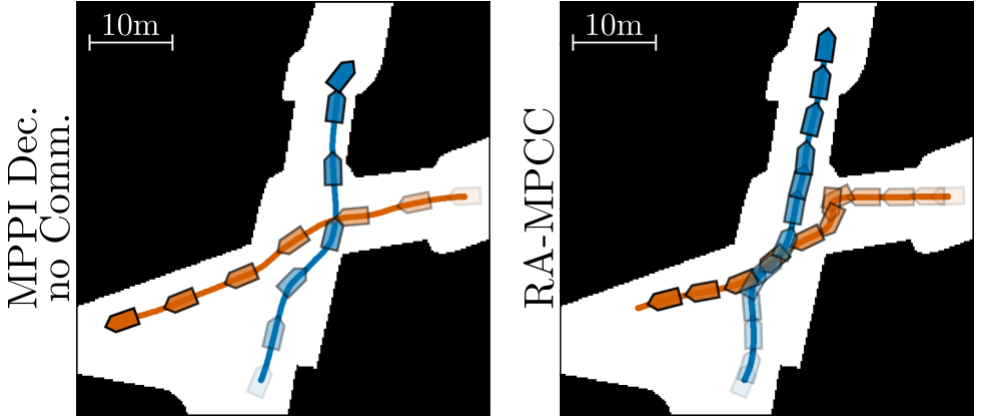
The proposed method is also capable of resolving scenarios with more than two agents. In Table 3.2 we show the results for 20 runs in crowded environments with four agents. The experiments are run in the maps shown in Fig. 3.7, where the vessels are exposed to many encounters in very narrow spaces. The results show that the centralized and decentralized method with shared local goals (with communication) can resolve the task successfully while behaving cooperatively. The decentralized version of our approach with no communication (thus predicting goals for other vessels) incurs in a collision in the tight Bloemgracht's intersection. With the four agents so close to each other, the vessels need to perform large avoidance maneuvers. This causes the estimated local goals and corresponding predictions to diverge drastically from the ground truth. However, similarly to the results in Table 3.1, decentralization has little effect on arrival time, total distance, and rule violations.



(a) *Prinsengracht*. MPPI understands that the blue vessel can safely cross in front of the vessel with the right of way (orange) without slowing it down. RA-MPCC is not confident and ends up blocking the way, pushing the orange vessel out of its route.



(b) *Bloemgracht*. The MPPI agents cooperate to perform collision avoidance, while RA-MPCC ends up in a deadlock.



(c) *Bloemgracht-Lijnbaansgracht*. The MPPI agents correctly solve the crossing according to the right of way. RA-MPCC, not understanding interactions, deems much of the state-space as occupied therefore steering left and right. Eventually, the agent with the right of way has to stop and pass behind, violating the navigation rules.

Figure 3.6: Comparisons between the decentralized MPPI without communications and RA-MPCC. Vessels on the figures are to scale (4m long) and are plotted every 5 seconds. Results for 100 runs are summarized in Table 3.1.

Table 3.1: Results for 100 runs of the experiments seen in Fig. 3.6 with randomized initial conditions and goals.

	Method	Successes- Deadlocks- Collisions	Rule Violations	Average Time	Total Average Distance
PG	Centralized	100 - 0 - 0	2	24.65s	82.15m
	Dec. Comm.	100 - 0 - 0	2	24.64s	82.14m
	Dec. No Comm.	100 - 0 - 0	2	25.12s	82.54m
	RA-MPCC	95 - 5 - 0	5	51.81s	73.20m
BG	Centralized	100 - 0 - 0	0	22.23s	74.93m
	Dec. Comm.	100 - 0 - 0	0	22.24s	74.79m
	Dec. No Comm.	100 - 0 - 0	0	22.18s	75.01m
	RA-MPCC	52 - 37 - 11	4	55.93s	67.64m
BGLG	Centralized	100 - 0 - 0	2	28.47s	87.24m
	Dec. Comm.	100 - 0 - 0	2	28.83s	88.26m
	Dec. No Comm.	100 - 0 - 0	2	29.01s	88.30m
	RA-MPCC	74 - 23 - 3	38	54.46s	84.20m

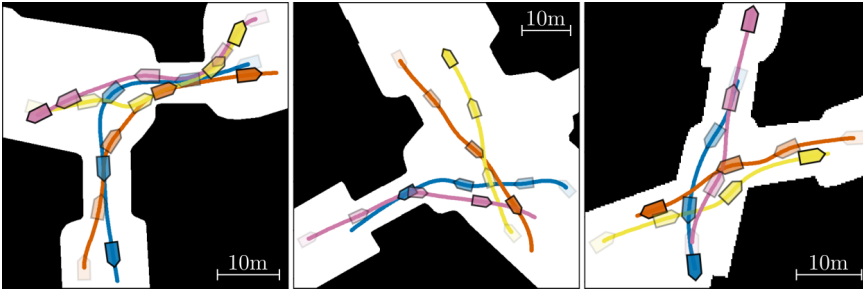


Figure 3.7: Navigation among four autonomous vessels running decentralized MPPI without communication. Left: Narrow intersection with a bridge in Herengracht. Middle: Wide intersection with bridges in Prinsengracht. Right: Very narrow intersection in Bloemgracht. Vessels are plotted every 6 seconds.

3.4.3 Navigation among Human-piloted Vessels

As long as the human-piloted agent behaves rationally, the resulting trajectories are very similar to the ones presented in Fig. 3.6, where all agents are autonomous. Therefore, in Fig. 3.8 we demonstrate how the proposed decentralized MPPI with no communication can cope with irrational agents.

3.4.4 Computational Complexity

As previously found in the literature [51], we confirm that MPPI scales linearly with an increasing number of agents (with constant sample number K). Table 3.3 shows that the algorithm runs at about 10Hz with two agents, therefore in real-time, and down to less than 4Hz with five agents. We want to stress that this was achieved with parallelization of the sampling procedure over the CPU (Intel® Xeon® W-2123 CPU @ 3.60GHz × 8, 64 GB).

Table 3.2: Results for 20 runs of the experiments seen in Fig. 3.7 with randomized initial conditions and goals.

Method		Successes- Deadlocks- Collisions	Rule Violations	Average Time	Total Average Distance
HG	Centralized	20 - 0 - 0	0	24.46	204.42
	Dec. Comm.	20 - 0 - 0	0	26.82	214.64
	Dec. No Comm.	20 - 0 - 0	3	33.18	225.48
PG	Centralized	20 - 0 - 0	0	23.24	214.92
	Dec. Comm.	20 - 0 - 0	0	23.82	216.99
	Dec. No Comm.	20 - 0 - 0	0	23.43	214.71
BGLG	Centralized	20 - 0 - 0	1	18.36	143.14
	Dec. Comm.	20 - 0 - 0	2	20.87	149.78
	Dec. No Comm.	19 - 0 - 1	0	20.13	144.86

Parallelizing this algorithm on a GPU would be highly beneficial, allowing for real-time control of several agents [51].

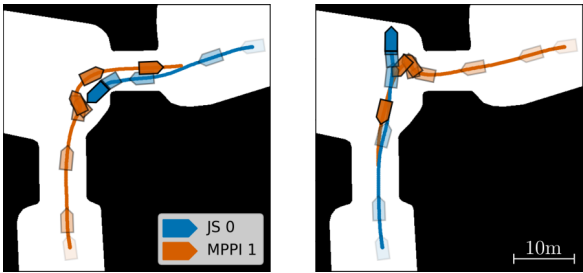


Figure 3.8: Qualitative robustness evaluation for non-cooperative vessels. Left: The joystick-driven vessel (blue) avoids to the wrong side. The MPPI vessel (orange) avoids collisions by coming to a complete stop, then continues to the left. Right: The joystick-driven vessel (blue) blocks the MPPI, disregarding right of way. The MPPI agent (orange) avoids collisions, comes to a stop, then continues on its way. Vessels are plotted every 4 seconds.

Table 3.3: Average computation time t_c and standard deviation $\sigma_{t,c}$ for increasing number of agents

Number of agents	2	3	4	5
t_c (ms)	90.7	137.2	182.9	209.9
σ_{t_c} (ms)	6.7	8.9	17.5	26.0

3.5 Conclusions

Within this work, we developed an MPPI controller for decentralized interaction-aware navigation in urban canals. In multiple sets of randomized scenarios, we demonstrate that our method outperforms a state-of-the-art MPC in terms of success rate, deadlocks, collisions, rule violations, and arrival times. In extensive experiments among several rational autonomous agents and case studies with potentially non-cooperative human drivers, we show robust operation while providing insights into the limitations of the approach. We display that decentralizing the MPPI does not sacrifice performance. Moreover, we demonstrate that the method would be able to run in real-time with multiple agents. In the future, however, a GPU implementation would vastly reduce the computation time. Moreover, the sampling distribution can be improved, thus requiring fewer samples to obtain a good approximation of the optimal control.

4

Learning-Based Predictions for Autonomous Surface Vessels

4

In Chapter 3, we introduced an Interaction-Aware Model Predictive Path Integral (IA-MPPI) controller that performs simultaneous prediction and planning in a decentralized, communication free setting. There we used a constant velocity model to estimate the hidden goal of the other non-communicating agents. In crowded, interaction-rich scenarios, agents have to significantly deviate from their path to avoid collisions with other agents, which can hinder the accuracy of the predicted goal.

In this Chapter, we use a centralized version of IA-MPPI [28] to generate an artificial dataset of trajectories featuring two to four Autonomous Surface Vessels (ASVs) navigating narrow urban intersections. We then adapt and train a trajectory prediction model and extract estimated goals from its predicted trajectories. We demonstrate that this approach can predict trajectories and extract goals more accurately than a constant velocity model, translating to better performances for the motion planner in interaction-rich environments.

Section 4.1 motivates our approach by providing a review of the literature and a contribution statement, while Section 4.2 gives a brief summary of IA-MPPI. In Section 4.3, we detail the prediction method, how the artificial data was generated, the training procedure, and the goal-extraction algorithm. Section 4.4 provides comparisons in simulation with an IA-MPPI that has access to the ground truth, that uses a constant velocity model, or that uses the proposed prediction algorithm. These results are summarized and conclusions are drawn in Section 4.5.

This chapter is a verbatim copy of the peer-reviewed paper [57]:

-  W. Jansma, E. Trevisan, A. Serra-Gómez and J. Alonso-Mora, “Interaction-Aware Sampling-Based MPC with Learned Local Goal Predictions,” 2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), Boston, MA, USA. 🏆 **Finalist for best paper award.**

Statement of contributions: Elia contributed to the initial idea of using a data-driven prediction model to extract goals for IA-MPPI [28] and provided theoretical insights. Under Elia and Alvaro’s supervision, Walter generated the artificial data and adapted and trained the Social-VRNN [58]. Elia, Walter, and Alvaro contributed equally to the experiments and the writing. Javier provided discussions and feedback, as well as editing of the manuscript.

4.1 Introduction

Cities characterized by dense networks of urban canals, such as Amsterdam, could greatly benefit from deploying ASVs for various tasks including deliveries, transportation of people, and garbage collection [54]. However, navigating autonomously in urban canals amidst mixed human-robot crowds presents a significant challenge. Urban canals are typically narrow, frequently congested, and lack the structured nature of roads. While not as strictly enforced as on roads, navigation principles like right-of-way and right-hand conventions should still be considered. Thus, akin to autonomous ground robots among pedestrian crowds, successful navigation in urban canals relies on cooperation and awareness of interactions [47].

4

Recently, a sampling-based Model Predictive Control (MPC) called IA-MPPI control has been developed for generating cooperative motion plans in urban canals among multiple non-communicating vessels while maintaining awareness of navigation rules [28]. This algorithm assumes rational and homogeneous agents, exact sensing of states, and knowledge of local goals. In real-time, the algorithm samples thousands of input sequences to approximate the optimal input sequence that enables all agents to progress toward their goals cooperatively. In scenarios where the local goals of other vessels are unavailable, such as in mixed human-robot environments or due to lack of communication, this previous approach has approximated these goals using a constant velocity model over a given horizon. However, in narrow and crowded environments, vessels often need to execute complex maneuvers to navigate tight intersections and avoid collisions while adhering to navigation rules. In such situations, relying solely on a constant velocity approximation can lead to inaccurate predictions, which can adversely affect the performance of the motion planner in terms of deadlocks, collisions, navigation rule violations, traveled distance, and travel time.

In this paper, we present a framework (see Fig. 4.1) that utilizes a learning-based trajectory prediction method to improve the estimation of agents' intended destinations. We introduce heuristics to extract local goals from the predicted trajectories and provide the motion planner with the flexibility to influence the behavior of other agents while expecting cooperation in collision avoidance.

4.1.1 Related Work

Robot motion planning in dynamic environments is a challenging problem for which a series of classical and heuristic-based approaches have been developed [59], such as the Dynamic Window Approach (DWA) [34] or Reciprocal Velocity Obstacles (RVO) [35, 36]. Despite their successful applications, e.g. to non-holonomic robots [60] or vessels in open waters [7], the motions planned by this class of methods are often reactive. This, especially in crowded environments, can lead to unsafe and unpredictable behaviors.

MPC has become a popular approach to trajectory planning for autonomous vehicles [61] because of its ability to optimize accounting for the system's dynamics and constraints. Moreover, by planning over a sufficiently large horizon, MPC can anticipate dynamic obstacles resulting in trajectories that are less reactive. To anticipate other agents, how-

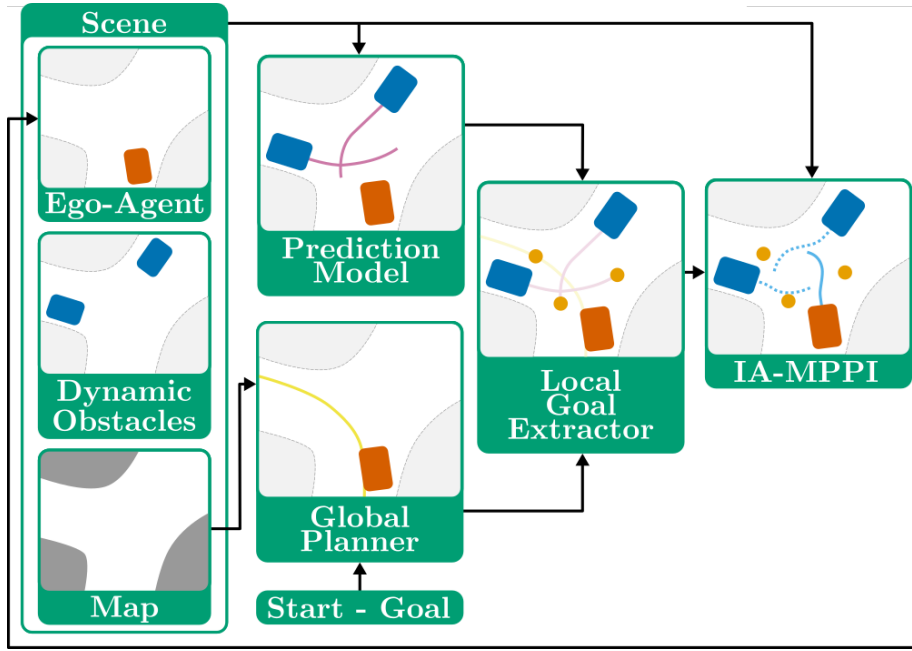


Figure 4.1: Overview of the proposed framework. Firstly, the prediction model utilizes information from all elements in the scene to forecast trajectories for obstacle agents. Meanwhile, the global planner, equipped with the map, start, and goal positions, generates a path for the ego agent. Subsequently, the local goal extractor leverages this information to determine appropriate local goals for the motion planner. With inputs derived from the scene and the local goals, the IA-MPPI algorithm simultaneously plans and predicts input sequences for all agents in the scene. The first input of the sequence is then assigned to the ego agent, and the algorithm iterates.

ever, the free space over the entire planning horizon needs to be computed [62], which requires knowledge about other agents' positions in the future. If all the agents in the environment are autonomous, communication and distributed optimization can be used to plan trajectories in multi-agent environments [44]. In mixed human-robot environments, however, such communication is not possible and predictions of the future motion of the other agents have to be employed. For instance, recent work on MPC for rule-aware navigation in urban canals uses constant velocity to model the future behavior of other vessels [8].

In interaction-rich scenarios, however, constant velocity can be an inaccurate approximation which may lead to unsafe motion plans [63]. Therefore, several works rely on learning-based models to predict the future motion of other agents [46] and can include prediction confidence [64] and multimodality [65]. These methods, however, decouple prediction and planning which, in high-interaction environments, may lead the ego agent to wrongly assume that no collision-free path exists [66]. To avoid the so-called freezing robot problem the robot has to expect cooperation in collision avoidance from the other agents [67]. Coupled prediction and planning can be done with MPC by modeling the in-

teracting agents as a system, but it quickly becomes expensive to solve via constrained optimization leading to long computation times and short planning horizons [48, 68].

Building upon a novel sampling-based MPC framework[10], IA-MPPI control [28] has successfully demonstrated decentralized coupled predictions and planning in real-time, accommodating long prediction horizons, nonlinear dynamics, and discontinuous cost functions in multi-agent environments. While IA-MPPI has exhibited superior performance compared to optimization-based MPC approaches that rely on fixed predictions of other agents' motion, it necessitates knowledge of their near-term local goals, which can either be communicated or estimated.

4.1.2 Contribution

This paper presents a novel framework for interaction-aware decentralized motion planning in urban canals without relying on communication. Our framework encompasses the following contributions:

- **Realistic Dataset:** We generate and publish a realistic dataset of simulated rule-abiding vessel trajectories in real sections of Amsterdam's urban canals.
- **Learning-Based Trajectory Prediction:** We adapt a pedestrian prediction model [58] to vessels and train it specifically for urban canals. This approach enables us to generate trajectory predictions for other agents.
- **Local Goal Extraction:** We propose heuristics to extract local goals from the predicted trajectories, thereby providing the motion planner with information about where agents intend to go.
- **Communication-Free Coupled Prediction and Planning:** By combining the local goal extraction with the IA-MPPI control [28], we achieve coupled prediction and planning without the need for communication. This approach ensures that the ego agent can influence the behavior of other agents while anticipating cooperation in collision avoidance.

We validate our planning framework through extensive simulated experiments, comparing it against baseline approaches and providing insights into the benefits of coupled prediction and planning over decoupled methods. The framework can be adapted to other robot types beyond vessels.

4.2 Interaction-Aware MPPI

In this section, we introduce the main ideas of IA-MPPI [28], upon which our proposed framework is built. For details on the method, models used and cost function please refer to the original paper. For insights on the underlying sampling-based MPC, one can refer to the work on Information-Theoretic MPC [10]. In short, IA-MPPI assumes that all the agents are homogenous and rational, i.e. have the same model and cost function. Under this assumption, we can create a large multi-agent system and plan input sequences resulting in cooperative trajectories for the ego agent as well as all the obstacle agents. This being a decentralized planning framework, we then apply the first input of the sequence

to our ego agent, observe the environment and plan again. In more detail, IA-MPPI models the ego-agent i as a discrete-time dynamical system,

$$\mathbf{q}_{i,t+1} = \mathcal{F}(\mathbf{q}_{i,t}, \mathbf{u}_{i,t}) \quad (4.1)$$

where $\mathbf{q}_{i,t}$ and $\mathbf{u}_{i,t}$ are, respectively, the state and the input of the ego-agent at timestep t . The state $\mathbf{q}_{i,t} = [\mathbf{p}_{i,t}, \mathbf{v}_{i,t}]$ contains the position and velocity of the agent. IA-MPPI assumes that all agents in the environment are homogenous. The state and the input of the multi-agent system consisting of the ego-agent and the obstacle agents can therefore be stacked, resulting in,

$$\begin{aligned} \mathbf{q} &= [\mathbf{q}_i^\top \quad \mathbf{q}_j^\top]^\top, \\ \mathbf{u} &= [\mathbf{u}_i^\top \quad \mathbf{u}_j^\top]^\top, \end{aligned} \quad \forall j \in \mathcal{M} \setminus i, \quad (4.2)$$

where $(\cdot)_j$ is a variable that the ego-agent i estimates of agent j and $\mathcal{M} = \{0, 1, \dots, m\}$ is the set of all agents in the scene. By also stacking the state transition functions $\mathcal{F}$ over all agents, we obtain a model for the multi-agent system $\mathbf{q}_{t+1} = \mathcal{G}(\mathbf{q}_t, \mathbf{u}_t)$. Given a planning horizon T and a prior input sequence $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{T-1}]$, IA-MPPI samples K input sequences for the entire multi-agent system,

$$\tilde{\mathbf{U}}_k = [\tilde{\mathbf{u}}_{0,k}, \tilde{\mathbf{u}}_{1,k}, \dots, \tilde{\mathbf{u}}_{T-1,k}], \quad \tilde{\mathbf{u}}_{t,k} = \mathcal{N}(\mathbf{u}_t, \nu \Sigma) \quad (4.3)$$

with $k = 1, \dots, K$, variance Σ and scaling parameter ν . At the first iteration, the prior input sequence $\mathbf{U}$ is initialized at zero. By the end of this section, it will become clear how this prior input sequence is updated in subsequent iterations. Having a model for the multi-agent system, we can forward simulate the K input sequences into K state trajectories $\mathbf{Q}_k$ for the multi-agent system,

$$\mathbf{Q}_k = [\mathbf{q}_0, \mathcal{G}(\mathbf{q}_0, \tilde{\mathbf{u}}_{k,0}), \dots, \mathcal{G}(\mathbf{q}_{k,T-1}, \tilde{\mathbf{u}}_{k,T-1})]. \quad (4.4)$$

Each of the resulting state trajectories is evaluated with respect to both an agent-centric cost as well as a system-wide cost, resulting in a total sample cost S_k . The reader can refer to the original publication for details on the cost function [28]. For the scope of our paper, it is important to know that the agent-centric cost includes a tracking cost to encourage progress towards a local goal p_g computed as,

$$C_{\text{tracking}} = k_{\text{tracking}} \frac{\|\mathbf{p}_g - \mathbf{p}_t\|_2}{\|\mathbf{p}_g - \mathbf{p}_{t_0}\|_2}, \quad (4.5)$$

where p_t is the position of the agent at timestep t , p_{t_0} is the position of the agent at the beginning of the planning horizon and k_{tracking} is a tuning parameter. Notice that we need to know the position of the local goal of each agent. For the ego agent, the local goal is extracted from a global plan. For all the other agents, the local goal has to be either communicated or estimated. We propose in the following section how this goal can be estimated. Once S_k , $\forall k \in [1, \dots, K]$ has been computed, importance sampling weights w_k can be calculated as,

$$w_k = \frac{1}{\eta} \exp\left(\frac{-1}{\lambda}(S_k - S_{\min})\right), \quad \sum_{k=0}^{K-1} w_k = 1, \quad (4.6)$$

where S_{min} is the minimum sampled cost, η a normalization factor and λ a tuning parameter. We then compute an approximation of the optimal control sequence through a weighted average of the sampled control sequences,

$$\mathbf{U}^* = \sum_{k=0}^{K-1} w_k \tilde{\mathbf{U}}_k \quad (4.7)$$

and apply the first input $\mathbf{u}_{i,0}^*$ to the ego-agent. We can now use a time-shifted version of $\mathbf{U}^*$ as the prior input sequence $\mathbf{U}$ to warm-start the sampling strategy at the next iteration.

4

4.3 Predicting Goal Positions

In Fig. 4.1 we provide an overview of the proposed framework. In Section 4.3.1, we outline the prediction model. In Section 4.3.2, we describe the dataset we have collected to train a prediction model that is interaction and rule-aware. In Section 4.3.3, we present the steps taken to port the prediction model to urban vessel environments. In Section 4.3.4, we propose a heuristic to extract a local goal suitable for IA-MPPI using the predicted trajectories.

4.3.1 Interaction-Aware Trajectory Prediction Method

Our approach leverages interaction-aware trajectory prediction for goal estimation. We employ an adapted version of *Social-VRNN* [58], which was originally designed for pedestrians, to obtain trajectory predictions. However, we remark that our framework is agnostic to the choice of trajectory predictor as long as it accounts for obstacles and interactions between agents in the environment.

Social-VRNN [58] is an interaction-aware trajectory prediction method that leverages a generative model based on Variational Recurrent Neural Networks (VRNNs) [69]. The model combines three types of contextual cues to define a joint representation of an agent's current state: information on the past trajectory of the agent of interest, environment context, and agent-agent interactions. The input to predict the trajectory of agent i is denoted as:

$$\mathbf{x} = \{\mathbf{v}_{-T_0:0}^i, \mathbf{O}_{env}^i, \mathbf{O}_{int}^{-i}\}, \quad (4.8)$$

where $\mathbf{v}_{-T_0:0}^i$ corresponds to the sequence of velocity states over the previous observed horizon T_O of the agent of interest i . The environment information $\mathbf{O}_{env}^i$ is represented in the form of a grid map extracted around the agent of interest. Then, $\mathbf{O}_{int}^{-i}$ represents the information on agent-agent interactions. It is a vector with the relative positions and velocities of all other agents from agent i 's perspective, listed in ascending order based on the absolute distance to it. The output of the model is a sequence of velocity probability distributions represented by T_H diagonal Gaussian distributions $\mathcal{N}(\mu_{\mathbf{v},k}, \text{diag}(\sigma_{\mathbf{v},k}^2))$. For details on the method and its architecture, please refer to the original paper [58].

Table 4.1: Specifications of the artificial dataset. *Exp.* refers to the number of experiments done in each scenario. *Frames* and *Vessels* refer to the total number of frames and vessels present in the data set, respectively. All data is recorded at a rate of 10Hz.

Scenario	Exp.	Frames	Vessels
Herengracht	1000	406229	2499
Prinsengracht	1247	420285	4122
Bloemgracht	1188	372173	3564
Open Crossing	1182	417515	3544
Amstel	79	23468	316
Total	4696	1639670	14045

4.3.2 Artificial Dataset

In the absence of a publicly available dataset for short-term vessel trajectory prediction, an artificial dataset of vessel interactions is collected in a simulation environment. In order to obtain trajectories that resemble those of real vessels in urban canals, four real canal section maps in Amsterdam: the Herengracht (HG), the Prinsengracht (PG) and the Bloemgracht (BG) are used to collect data. The Open Crossing (OC) environment is created to collect vessel interactions in open water. Data on an additional environment, the Amstel (AM), is included only for testing our framework’s generalization to environments not seen during training. Figure 4.2 depicts two of these canal sections. The yellow rectangles correspond to the areas in which start and goal locations are randomly initialized. These areas are placed around the entire map and in each canal section to improve the diversity of the trajectories and interactions.

To collect the data, more than four thousand experiments are conducted by initializing up to four vessels simultaneously in the mentioned environments. Each vessel is assigned a randomized start and goal location in one of the predefined areas. All sampled locations are ensured to be collision-free. The vessels run a centralized IA-MPPI to sail toward their respective goals while accounting for navigation rules. This ensures that the recorded trajectories are safe, interaction-aware, and mostly rule-abiding.

For each experiment and vessel in the environment we record the current timestamp, the vessel ID, its position and velocity in the global frame. Each timestamp is unique across timesteps and experiments, which enables to identify vessels belonging to the same scene. The specifications of the artificial vessel dataset can be found in Table 4.1. In order to evaluate the prediction model, 10% of the dataset is used as the test set. The remaining data is used for training and is split into a training set (72%) and a validation set (18%). The distribution of data from each scenario is ensured to be equal in all splits.

4.3.3 Model Training and Adaptation

We adapt the variational inference architecture presented in [58] to generate unimodal trajectory probability predictions of vessels. In contrast to humans, vessels are slower and have lower-order dynamics, which results in less reactive behaviors and smoother trajectories. To take this into account and avoid overfitting to the dataset, we reduce the

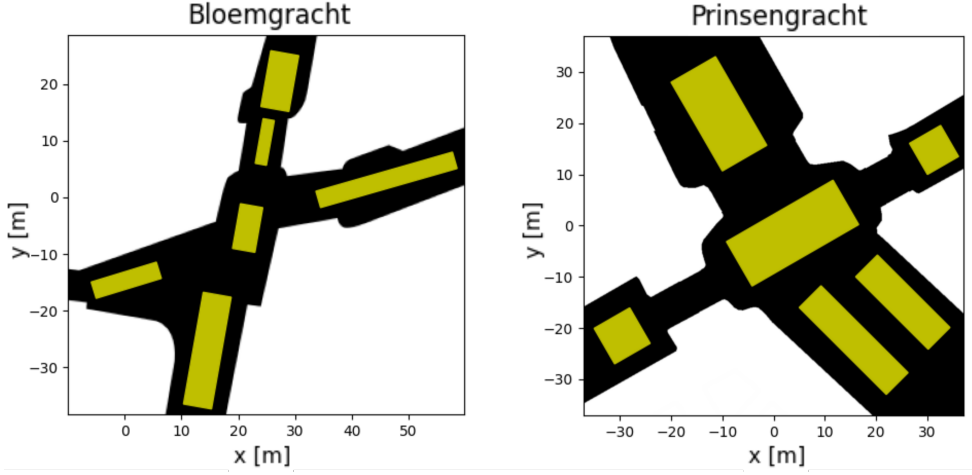


Figure 4.2: Canal sections of the Bloemgracht and Prinsengracht. The black areas are the canals. The yellow rectangles correspond to the initialization areas in which goals and starting locations were randomly initialized for each agent during the simulations.

dimensionality of the method's latent space. We also add an L2-regularization term to the loss function and weight it with a hyperparameter we define as γ .

Hyperparameters

The model is trained using backpropagation through time and the RMSProp [70] optimizer. With a time step of $\Delta T = 0.4$ seconds, the prediction horizon is set to $T_H = 24$ steps (9.6 seconds) and the previous horizon to $T_O = 14$ steps (5.6 seconds). Furthermore, we employ learning rate starting at $\alpha = 1e-4$ that decays by a factor of 0.9 after every gradient step. The regularization weight is kept at $\gamma = 0.0001$. Finally, the model is trained for $4e4$ training steps, using early stopping.

4.3.4 Local Goal Extraction

In eq. (4.5) we show that the IA-MPPI needs to know the local goal p_g of each agent. There are two requirements for a goal to be suitable: it has to lie within a radius r_{p_g} from the agent it corresponds to and cannot be in space occupied by static obstacles. Therefore, we first search the predicted trajectory backward until we obtain a position $p_{\leq r_{p_g}}$ within the desired radius. If $p_{\leq r_{p_g}}$ is in collision with a static obstacle, we construct a circle centered on the agent's position p_a with radius $p_a - p_{\leq r_{p_g}}$ and find the point on the circle closest to $p_{\leq r_{p_g}}$ which is not in collision with static obstacles. This goal extraction method is illustrated in Fig. 4.3. Once the goals for all agents are predicted, IA-MPPI can plan interaction-aware trajectories in a decentralized fashion.

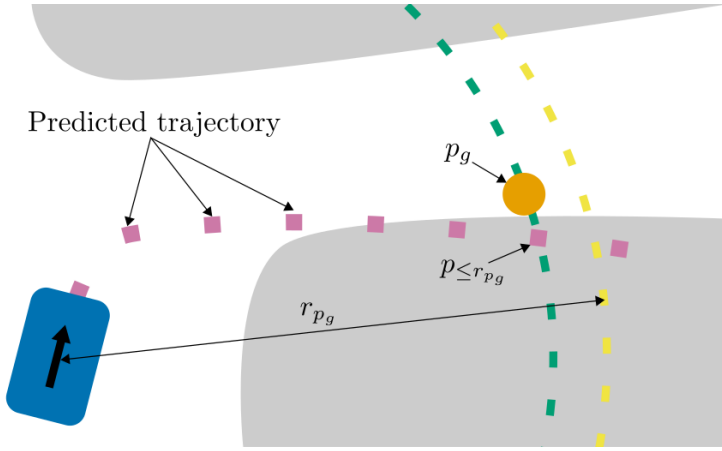


Figure 4.3: A visual illustration of how the local goal is extracted from a colliding trajectory prediction.

4.4 Experiments

The experiments are conducted in real maps of Amsterdam’s canals, namely the Herengracht (HG), Bloemgracht (BG), Prinsengracht (PG), and the Amstel (AM). In addition, experiments are conducted in an Open Crossing (OC) map without static obstacles. In Section 4.4.1 we evaluate the prediction model, in Section 4.4.2 we show the performances of the proposed framework for motion planning, and in Section 4.4.3 we highlight the benefits of coupled prediction and planning with respect to a decoupled approach.

4.4.1 Prediction Accuracy

In Fig. 4.4 we compare the proposed Learning-Based Model (LBM) to a Constant Velocity Model (CVM) on test data. We evaluate the methods against the displacement error at each prediction step, which is defined as the Euclidean distance between a prediction and the ground truth. In all maps the LBM outperforms the CVM, showing a lower average displacement error and a smaller standard deviation. Note that the Amstel map was previously unseen during training, demonstrating generalization capabilities.

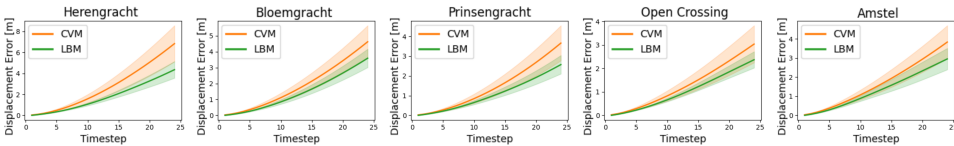


Figure 4.4: The displacement error of the predictions from CVM and the LBM over the prediction horizon for each canal section. The solid line represents the mean error and the shaded area represents 30% of the standard deviation.

4.4.2 Interaction-Aware Motion Planning with Predictions

In this study, we evaluate the performance of the proposed decentralized framework that uses a Learning-Based prediction Model to extract local goals (IA-MPPI-LBM), by comparing it against a decentralized approach that extracts local goals from a Constant Velocity Model (IA-MPPI-CVM) and decentralized with communication (IA-MPPI-w/comm.), which assumes perfect knowledge of other agents' local goals. It is important to stress that, in similar experiments, the IA-MPPI-CVM which serves as the communication-free baseline in our comparisons has already been demonstrated to outperform an optimization-based MPC approach that relies on fixed predictions [28].

In the simulated experiments taking place in real sections of the canals of Amsterdam, we randomize the initial positions and goals of four interacting agents, all running the same algorithm. To challenge each method, we design regions within which each agent's start and goal position are randomly initialized in a way that forces all four agents to interact in a narrow section of the map. These *high-interaction* scenarios are discussed in Section 4.4.2.

For completeness, we also design experiments where agents' starting and goal positions are randomized across much larger spaces. In these experiments, however, vessels don't often interact and usually have larger free spaces to avoid each other. These *low-interaction* scenarios are discussed in Section 4.4.2.

An example of experiments in low- and high-interaction scenarios is shown in Fig. 4.5. The IA-MPPI plans with a time horizon T of 100 time steps with step size $\delta T = 0.1s$ and $K = 4500$ samples. Each method is evaluated on the same set of randomly initialized experiments. For fairness, metrics such as rule violations, goal displacement error, total traveled distance, and time are only displayed for experiments that ended successfully with all methods.

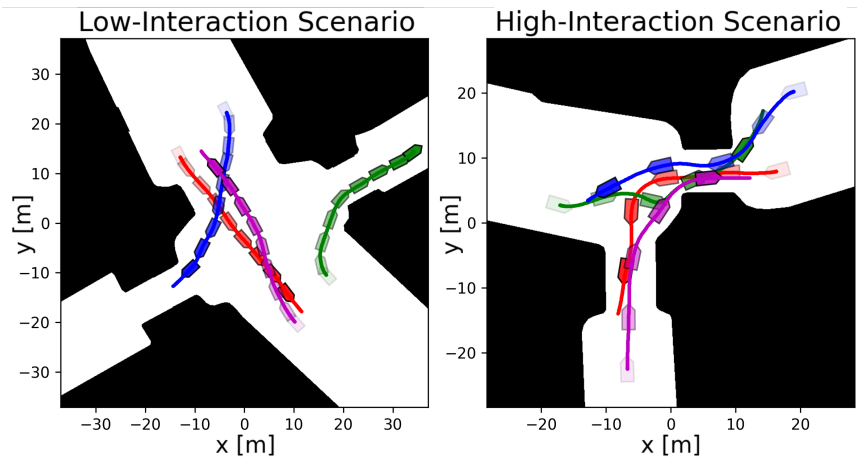


Figure 4.5: Examples of experiments in the low-interaction scenario (left) and high-interaction scenario (right).

High-Interaction Scenario

The experiments in high-interaction scenarios are conducted in narrow intersections in the Bloemgracht, Herengracht, and Prinsengracht. Since the Amstel canal is very wide and the Open Crossing has no static map constraints, it is difficult to generate experiments with high-interactions, and thus these two maps are excluded from this experiment section. The results of the experiments are summarized in Table 4.2 and Figure 4.6. It can be seen that in these high-interaction scenarios, the LBM consistently outperforms the CVM in terms of the goal displacement error (Goal DE). As a consequence, the motion planning framework that estimates other agents' local goals using predictions from the LBM outperforms the framework that uses the CVM on all the metrics. Moreover, we demonstrate our framework with the LBM has negligible performance losses compared to the method with perfect communication.

Table 4.2: Successes (Succ.), Deadlocks (Deadl.), Collisions (Coll.), Rule Violations (Rule Viol.) and Goal Displacement Error (Goal DE) for all methods in high-interaction scenarios per canal sections.

	Method	Succ. / Deadl. / Coll.	Rule Viol.	Goal DE
HG	IA-MPPI-CVM	18 / 0 / 2	16	5.82 m
	IA-MPPI-LBM (ours)	19 / 1 / 0	16	5.26 m
	IA-MPPI-w/comm.	20 / 0 / 0	16	
PG	IA-MPPI-CVM	19 / 0 / 1	11	7.30 m
	IA-MPPI-LBM (ours)	19 / 1 / 0	5	4.11 m
	IA-MPPI-w/comm.	20 / 0 / 0	5	
BG	IA-MPPI-CVM	17 / 0 / 3	7	4.98 m
	IA-MPPI-LBM (ours)	20 / 0 / 0	4	4.13 m
	IA-MPPI-w/comm.	20 / 0 / 0	3	

Low-Interaction Scenarios

Table 4.3 and Fig. 4.7 summarize the results in low-interaction scenarios. Note that we here also test on the Open Crossing maps and the Amstel, which the LBM has not previously seen in training. The results show that also when the start and goal positions of all agents are randomly initialized over large areas, our proposed communication-free framework with the LBM performs just as well as the baseline with full communication, even in a map unseen in training. However, perhaps unsurprisingly, the framework that approximates the local goals with a CVM can also achieve the same performance as the framework with full communication. Intuitively, in low-interaction scenarios where agents mostly navigate straight to their goal, CVM is a reasonably good approximator.

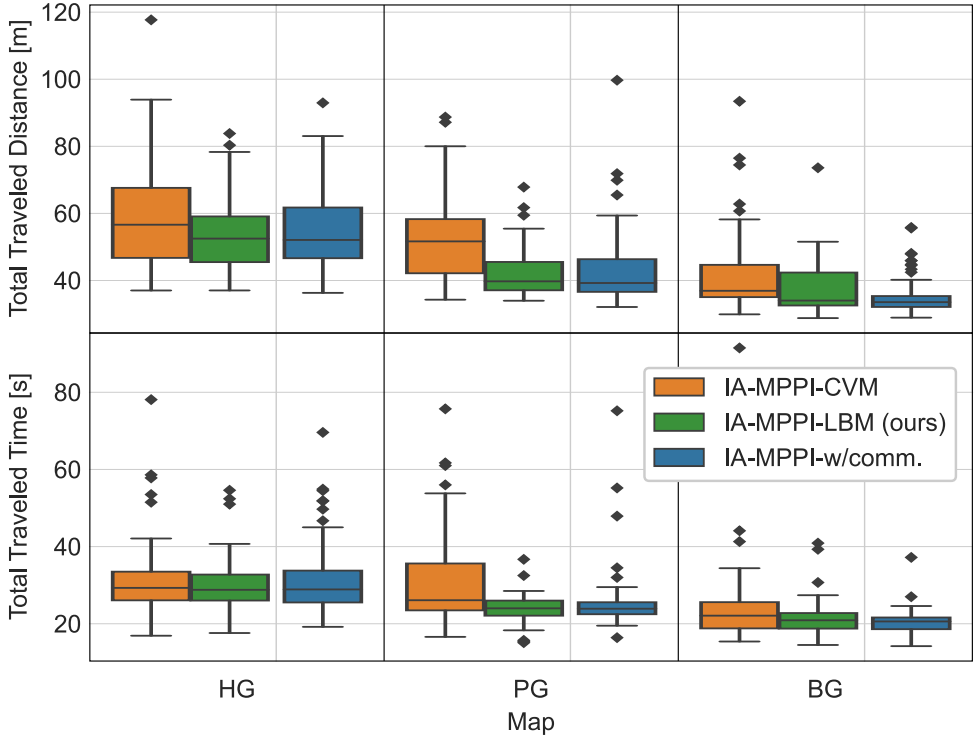


Figure 4.6: This figure displays the distribution of the total traveled distance and total traveled time of the vessels during the experiments in the high-interaction scenarios. The results are displayed per map and for each method.

4.4.3 Decoupled Prediction and Planning

The framework we proposed utilizes a LBM to predict trajectories for obstacle agents and extract local goals while employing IA-MPPI for coupled predictions and planning. To assess the advantages of this framework, we compare it to a planner without interaction awareness (MPPI-LBM), which decouples prediction and planning. Like other state-of-the-art methods, MPPI-LBM treats the predicted future trajectories of obstacle agents as occupied space and plans the ego agent's motion without considering interaction awareness. This approach reduces the system size and computational burden by minimizing the space to be sampled. However, apart from this difference, MPPI-LBM shares the same sampling strategy and cost function as the proposed IA-MPPI-LBM. We conducted 100 low-interaction experiments across the Amstel, Bloemgracht, Herengracht, Open Crossing, and Prinsengracht, comparing different methods. Table 4.4 presents the outcomes, including total successes, deadlocks, collisions, and rule violations. Again, the proposed IA-MPPI-LBM shows similar performances to the method with communication (IA-MPPI-w/comm).

However, MPPI-LBM exhibited a significantly lower success rate and a higher number of rule violations. The LBM, while trained to be somewhat rule- and interaction-aware in its predictions, occasionally struggles to capture complex reciprocal collision avoidance

Table 4.3: Successes (Succ.), Deadlocks (Deadl.), Collisions (Coll.), Rule Violations (Rule Viol.), and Goal Displacement Error (Goal DE) for all methods in the various canal sections.

	Method	Succ. / Deadl. / Coll.	Rule Viol.	Goal DE
HG	IA-MPPI-CVM	38 / 0 / 2	22	5.31 m
	IA-MPPI-LBM (ours)	37 / 0 / 3	26	5.42 m
	IA-MPPI-w/comm.	37 / 2 / 1	26	
PG	IA-MPPI-CVM	38 / 0 / 2	14	3.12 m
	IA-MPPI-LBM (ours)	38 / 0 / 2	13	2.94 m
	IA-MPPI-w/comm.	38 / 0 / 2	14	
BG	IA-MPPI-CVM	38 / 0 / 2	17	5.36 m
	IA-MPPI-LBM (ours)	39 / 0 / 1	20	4.93 m
	IA-MPPI-w/comm.	40 / 0 / 0	20	
OC	IA-MPPI-CVM	40 / 0 / 0	27	3.98 m
	IA-MPPI-LBM (ours)	40 / 0 / 0	29	4.04 m
	IA-MPPI-w/comm.	40 / 0 / 0	27	
AM	IA-MPPI-CVM	40 / 0 / 0	22	3.12 m
	IA-MPPI-LBM (ours)	39 / 1 / 0	18	3.49 m
	IA-MPPI-w/comm.	40 / 0 / 0	21	

maneuvers when agents are in close proximity. This, combined with the motion planner's unawareness of the ego agent's influence on other agents' motion and their cooperation in collision avoidance, often led the MPPI-LBM to wrongly assume that no feasible solution existed. Consequently, this resulted in agents drifting into collisions due to their large inertia.

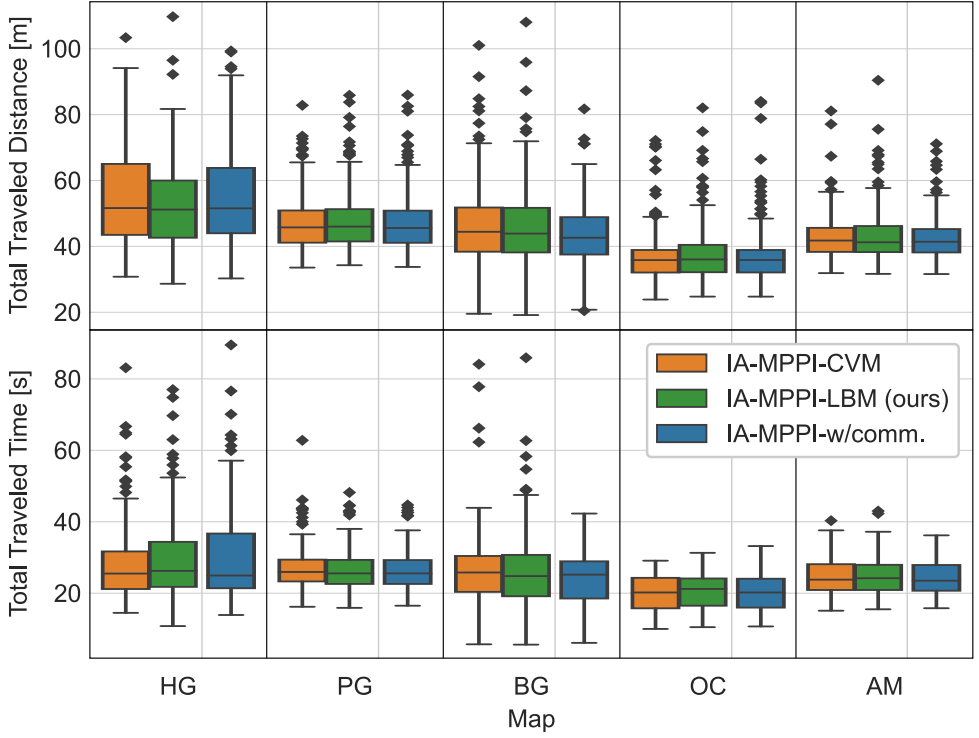


Figure 4.7: This figure displays the distribution of the total traveled distance and total traveled time by vessels during the experiments in the low-interaction scenarios. The results are displayed per map and for each method.

Table 4.4: Successes (Succ.), Deadlocks (Deadl.), Collisions (Coll.), and Rule Violations (Rule Viol.) for the non-interactive MPPI and the IA-MPPI baseline in low-interaction scenarios.

Method	Succ. / Deadl. / Coll.	Rule Viol.
MPPI-LBM.	72 / 1 / 27	40
IA-MPPI-LBM (ours)	97 / 1 / 2	30
IA-MPPI-w/comm.	98 / 0 / 2	28

4.5 Conclusions

In this paper, we introduced a framework that combines a learning-based trajectory prediction model with IA-MPPI, enabling decentralized and communication-free coupled prediction and planning. Our experimental results demonstrated the superiority of our LBM over the CVM in accurately predicting the trajectories of interacting vessels, even in unseen maps. Through simulated experiments in Amsterdam’s canals, we showed that our motion planning framework achieved comparable performance to a method with ground

truth knowledge of local goals, which was shown to outperform classical optimization-based MPC approaches with decoupled prediction and planning in previous work [28]. Additionally, we highlighted the limitations of the CVM in tight environments with multiple interacting agents. Finally, by comparing our approach with a non-interactive planner, we emphasized the advantages of coupled planning and predictions.

5

Informing the Motion Planner with Ancillary Controllers

5

In Chapter 3 and Chapter 4, we introduced a decentralized, communication-free, interaction-aware Model Predictive Path Integral control (MPPI) controller for simultaneous prediction and planning while accounting for discontinuous navigation rules. Although effective, this method sometimes fails to react swiftly to unexpected events or abrupt environmental changes because classic MPPI only samples around the previous control sequence.

To enhance MPPI's reactivity and efficiency, this chapter introduces ancillary controllers into the sampling distribution. This adjustment biases the sampling distribution, making MPPI more responsive and efficient. By incorporating these ancillary controllers, we improve the controller's ability to handle sudden changes and unforeseen obstacles, ensuring safer and more reliable navigation in dynamic urban canal environments.

Section 5.1 provides further motivation, a brief literature review, and our contributions. Section 5.2 covers the necessary preliminaries. Our proposed approach, detailing the integration of ancillary controllers into the MPPI framework, is described in Section 5.3. An illustrative example using a pendulum experiment is presented in Section 5.4 to visualize the samples and the effect of the ancillary controllers. Section 5.5 provides motion planning experiments and results with simulated boats in canals and a real-robot experiment using the Jackal robot. Finally, the chapter concludes with a summary of findings in Section 5.7.

This chapter is a verbatim copy of the peer-reviewed paper [71]:

-  **E. Trevisan** and J. Alonso-Mora, "Biased-MPPI: Informing Model Predictive Path Integral Control by Fusing Ancillary Controllers," *IEEE Robotics and Automation Letters (RA-L)*, May 2024.

Statement of contributions: Elia contributed to the initial idea of using ancillary controllers to inform the sampling distribution, developed the theory, performed both the simulated and real-robot experiments, and was the main author of the manuscript. Javier provided discussions on the ideas and feedback, as well as editing of the manuscript.

5.1 Introduction

Navigating autonomous robots through dense and dynamic environments poses a formidable challenge due to significant uncertainties, including the robot's state, model, environmental conditions, and interactions with other agents. Achieving desired behaviors under such conditions often necessitates using intricate cost functions and constraints, resulting in complex, nonlinear, non-convex, and occasionally discontinuous problem formulations. The dynamic nature of the environment introduces potential unexpected changes, demanding rapid adaptability in the robot's actions.

To address these challenges, one approach is to cast the problem in a stochastic optimal control setting, where they can be mathematically represented as stochastic Hamilton-Jacobi-Bellman (HJB) equations. However, solving these equations numerically can be challenging due to the curse of dimensionality. Pioneering work demonstrated that the stochastic HJB equations can be linearized for control-affine systems, and their solution can be approximated through sampling using the path integral formulation [13]. Implemented in a receding horizon fashion, MPPI control [25, 51], and its Information-Theoretic counterpart [10, 21] have been initially used for racing a small-scale rally car. MPPI has also been successfully applied to several other planning problems, such as for autonomous vehicles with dynamic obstacles [72], solving games [73], flying drones in partially observable environments [74], performing complex maneuvers [75] and used in combination with adaptive control schemes [76]. It has also been adapted to multi-agent systems for formation flying [77], cooperative behavior [78], and simultaneous prediction and planning [28]. Furthermore, MPPI has shown promise in manipulating objects with robot arms [79] including model uncertainties [80], in pushing tasks [81, 82] and planning motion for four-legged walking robots [83]. MPPI is a model-based approach that requires a model to forward simulate trajectories given sampled inputs. Recent work has utilized physics engines to simulate samples [84, 85], eliminating the need for explicitly defining the dynamics of agents and the environment, thus providing a significant advantage in contact-rich manipulation tasks.

One of the critical challenges in applying MPPI to dynamic environments is ensuring the algorithm's performance and reliability. The success of MPPI heavily relies on the choice of sampling distribution, which is crucial, especially in real-time scenarios. Most existing literature uses the previously computed input sequence as the mean of a Gaussian distribution for sampling [25]. However, using the previous input sequence may trap the algorithm in local minima and can lead to catastrophic failures in the presence of unexpected disturbances or changes in the environment [86] (Figure 5.1). This paper explores the application of MPPI in dynamic environments, emphasizing the need to improve its performance and reliability in the face of unexpected disturbances and rapidly changing conditions.

5.1.1 Previous Work

Several works tried to make the method more efficient or more robust. Early work [87] proposed using Expectation Propagation instead of Monte Carlo sampling, demonstrating better efficiency in scenarios with hard constraints. Other works instead accelerate the convergence of MPPI by leveraging gradient descent updates [88]. Another option to be

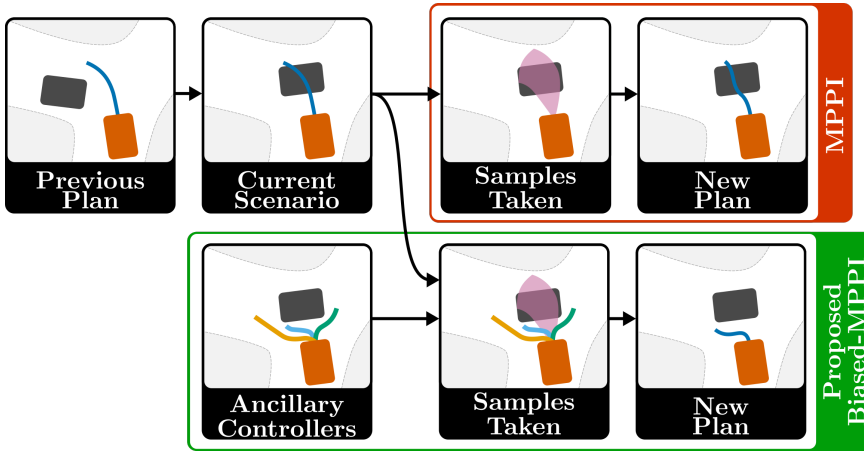


Figure 5.1: *Top*: Usually, MPPI only takes samples around a previous plan. Here, the environment changes unexpectedly, and all the sampled trajectories are in collision, which leads to computing a new plan that also collides. *Bottom*: our Biased-MPPI adds ancillary controllers to the sampling distribution, quickly converging to a collision avoidance maneuver.

more reactive to environmental changes is to iteratively converge to a solution through adaptive importance sampling [89]. This, however, requires multiple iterations between each planning time step, diminishing the parallelizability of MPPI. Many other works propose improving the algorithm's convergence by somehow changing its sampling distribution. This can be done by substituting the Gaussian used for sampling with a different hand-crafted distribution [90] or by directly learning a distribution from data [91, 92]. Given that MPPI allows for tuning the variance of the sampling distribution [51], some works sought to improve the efficiency of the scheme by adapting the covariance online via covariance steering [93, 94]. Other ways to improve efficiency can be to fit splines to the sampled inputs [79] or to constrain the distribution to sample areas that are known to contain low-cost trajectories [83]. Previous works have also experimented with ancillary controllers. In [95], authors propose to sample inputs around a path previously computed by Rapidly exploring Random Tree (RRT). Other works instead robustify MPPI by switching to an iterative Linear Quadratic Gaussian (iLQG) controller [86] or by integrating one into the system's model [96]. Previous work also compares an MPPI that samples around a previously computed input, an input sequence computed by a sequential linear-quadratic Model Predictive Control (MPC), and a learned sampling policy [83]. In general, however, the original derivations of MPPI [10] only allow samples to be drawn from a uni-modal Gaussian distribution, usually centered around the previous control sequence, which can hamper performance and reduce reactivity to unexpected changes in the environment.

5.1.2 Contributions

We propose a Biased-MPPI, for which we provide mathematical derivations that allow for arbitrary changes to the sampling distribution. We discuss the impact of introducing

biases in the sampling distribution on the overall method. We experiment with an importance sampler that utilizes multiple classical and learning-based ancillary controllers simultaneously to take more informative samples, which can be seen as a control fusion scheme. Through simulated and real-world experiments, we demonstrate the impact of taking suggestions from several underlying controllers on robustness to model uncertainties and local minima, reactivity to unexpected events, and sampling efficiency.

5.2 Preliminaries

In this section, we provide a concise introduction to the key concepts of MPPI within the Information-Theoretic framework. For more details, we direct the reader to prior research [10]. We begin by defining a function:

$$\mathcal{F}(S, P, x_0, \lambda) = -\lambda \log \left(\mathbb{E}_P \left[\exp \left(-\frac{1}{\lambda} S(V) \right) \right] \right) \quad (5.1)$$

which we will denote as the free energy of the system. Here, V represents a sequence of inputs, P is a base measure, λ is a tuning parameter, $S(V)$ is a cost, and x_0 represents the system's initial state. It can be shown that:

$$\mathcal{F}(S, P, x_0, \lambda) \leq \mathbb{E}_Q[S(V)] + \lambda \text{KL}(Q||P). \quad (5.2)$$

Here, Q represents a probability measure that characterizes the controlled input distribution, and $\text{KL}(Q||P)$ denotes the Kullback–Leibler (KL)-Divergence between the base measure and the controlled measure. equation (5.2) signifies that the free energy serves as a lower bound for the expected cost under the controlled distribution plus a control cost represented by the KL-Divergence. Hence, determining a control distribution that achieves this lower bound minimizes the expected cost and control cost. We can define a control distribution Q^* through its Radon-Nikodym derivative to the base measure:

$$\frac{dQ^*}{dP} = \frac{\exp(-\frac{1}{\lambda} S(V))}{\mathbb{E}_P[\exp(-\frac{1}{\lambda} S(V))]} \quad (5.3)$$

Substituting Q with Q^* in equation (5.2), we can prove that Q^* is an optimal control distribution in the sense that it achieves the lower bound. The idea is now to align our control distribution Q with the optimal distribution Q^* though KL minimization, which results in the optimal input sequence U^* :

$$U^* = \underset{U}{\operatorname{argmin}} \text{KL}(Q^*||Q). \quad (5.4)$$

Now, considering a discrete-time system:

$$x_{t+1} = F(x_t, v_t), \quad v_t \sim \mathcal{N}(u_t, \Sigma). \quad (5.5)$$

Here, $x_t \in \mathbb{R}^n$ represents the state vector at time step t , $F(\cdot)$ is the state transition model, $v_t \in \mathbb{R}^m$ denotes the noisy input, $u_t \in \mathbb{R}^m$ is the commanded input, and Σ corresponds to the

natural input variance of the system. If P and Q are the uncontrolled and controlled measures, respectively, we can define them through their probability density functions:

$$p(V) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m |\Sigma|)^{1/2}} \exp\left(-\frac{1}{2} v_t^T \Sigma^{-1} v_t\right)$$

$$q(V|U) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m |\Sigma|)^{1/2}} \exp\left(-\frac{1}{2} (v_t - u_t)^T \Sigma^{-1} (v_t - u_t)\right).$$

It can be proven from equation (5.4) that the optimal control input at time t is the mean input under the optimal distribution:

$$u_t^* = \int_{\Omega_V} q^*(V) v_t dV. \quad (5.6)$$

We can estimate such mean sampling from our controlled distribution via importance sampling:

$$u_t^* = \int \frac{q^*(V)}{q(V|U)} q(V|U) v_t dV$$

$$= \mathbb{E}_Q[\omega(V) v_t], \quad (5.7)$$

with the importance sampling weight $\omega(V)$ being:

$$\omega(V) = \left(\frac{q^*(V)}{q(V|U)} \right) = \left(\frac{q^*(V)}{p(V)} \right) \left(\frac{p(V)}{q(V|U)} \right)$$

$$= \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} \left(S(V) + \frac{\lambda}{2} \sum_{t=0}^{T-1} u_t^T \Sigma^{-1} u_t + 2 u_t^T \Sigma^{-1} \epsilon_t \right)\right). \quad (5.8)$$

We can, therefore, sample K noisy input sequences:

$$V^k = [v_0^k, v_1^k, \dots, v_t^k, \dots, v_{T_H}^k]$$

$$v_t^k \sim \mathcal{N}(u_t, \Sigma) \quad (5.9)$$

where t is a time step and T_H is the planning horizon. A practical choice often made in MPPI is to take u_t as a time-shifted version of the previously computed approximation of the optimal control sequence. We roll out the sampled V^k into state trajectories using the system's model $F(\cdot)$, evaluate their cost $S(V)$, compute the weights $\omega(V)$, get a new estimate of the optimal input sequence U^* via equation (5.7) and iterate. In equation (5.8), the control cost is multiplied and divided by λ . Not having control over the magnitude of the terms at the exponential can cause numerical issues. A change of base measure $\mathbb{P}$ can solve the problem [10]. One might also need a higher variance Σ_s for sampling compared to the natural variance of the system Σ [12]. This again introduces terms at the exponential independent from λ . Moreover, introducing an arbitrary, potentially multi-modal sampling distribution Q_s is difficult. All these issues stem from the ratio $p(v)/q(V|U)$ in equation (5.8). Our approach addresses this by showing that accepting a bias in the solution can eliminate the ratio and allow for arbitrary sampling distributions.

5.3 Proposed Approach

5.3.1 Biased-MPPI

Let us first redefine the cost function as:

$$\tilde{S}(V) = S(V) + \lambda \log \left(\frac{p(V)}{q_s(V)} \right). \quad (5.10)$$

We then define the free-energy with this new cost:

$$\begin{aligned} \mathcal{F}(\tilde{S}, P, x_0, \lambda) &= -\lambda \log \left(\mathbb{E}_P \left[\exp \left(-\frac{1}{\lambda} \tilde{S}(V) \right) \right] \right) \\ &= -\lambda \log \left(\mathbb{E}_Q \left[\exp \left(-\frac{1}{\lambda} \tilde{S}(V) \right) \frac{p(V)}{q(V)} \right] \right) \\ &\leq -\lambda \mathbb{E}_Q \left[\log \left(\exp \left(-\frac{1}{\lambda} \tilde{S}(V) \right) \frac{p(V)}{q(V)} \right) \right] = * \end{aligned} \quad (5.11)$$

where, as in [10], we applied Jensen's inequality. We can simplify the right-hand side as follows:

$$\begin{aligned} * &= -\lambda \mathbb{E}_Q \left[-\frac{1}{\lambda} \tilde{S}(V) + \log \left(\frac{p(V)}{q(V)} \right) \right] \\ &= -\lambda \mathbb{E}_Q \left[-\frac{1}{\lambda} S(V) - \log \left(\frac{p(V)}{q_s(V)} \right) + \log \left(\frac{p(V)}{q(V)} \right) \right] \\ &= \mathbb{E}_Q [S(V)] + \lambda \mathbb{E}_Q \left[\log \left(\frac{p(V)}{q_s(V)} \frac{q(V)}{p(V)} \right) \right] \\ &= \mathbb{E}_Q [S(V)] + \lambda \text{KL}(Q \| Q_s). \end{aligned}$$

The free energy inequality is then:

$$\mathcal{F}(\tilde{S}, P, x_0, \lambda) \leq \mathbb{E}_Q [S(V)] + \lambda \text{KL}(Q \| Q_s). \quad (5.12)$$

Thus, while we start with $\tilde{S}(V)$, the free energy serves as a lower bound for the expected original cost $S(V)$ under the controlled distribution plus lambda times the KL-Divergence between the controlled and sampling distribution. An optimal control distribution achieving the lower bound would minimize the original cost $S(V)$ while pushing the controlled distribution to align with the sampling distribution, effectively introducing a bias toward the sampling distribution. We define a controlled distribution Q^* as:

$$\frac{dQ^*}{dP} = \frac{\exp(-\frac{1}{\lambda} \tilde{S}(V))}{\mathbb{E}_P[\exp(-\frac{1}{\lambda} \tilde{S}(V))]}.$$

Under Q^* , the KL-Divergence becomes:

$$\begin{aligned}
 \text{KL}(Q^* \| Q_s) &= \mathbb{E}_{Q^*} \left[\log \left(\frac{q^*(V)}{q_s(V)} \right) \right] \\
 &= \mathbb{E}_{Q^*} \left[\log \left(\frac{q^*(V)}{p(V)} \right) \right] + \mathbb{E}_{Q^*} \left[\log \left(\frac{p(V)}{q_s(V)} \right) \right] \\
 &= -\frac{1}{\lambda} \mathbb{E}_{Q^*} [\tilde{S}(V)] - \log \left(\mathbb{E}_P \left[\exp \left(-\frac{1}{\lambda} \tilde{S}(V) \right) \right] \right) \\
 &\quad + \mathbb{E}_{Q^*} \left[\log \left(\frac{p(V)}{q_s(V)} \right) \right]
 \end{aligned}$$

Substituting into equation (5.12) and simplifying leads to:

$$\begin{aligned}
 \mathcal{F}(\tilde{S}, P, x_0, \lambda) &\leq -\lambda \log \left(\mathbb{E}_P \left[\exp \left(-\frac{1}{\lambda} \tilde{S}(V) \right) \right] \right) \\
 &= \mathcal{F}(\tilde{S}, P, x_0, \lambda).
 \end{aligned}$$

This proves that Q^* is the optimal distribution in that it achieves the lower bound in equation (5.12). Following the steps in [10], we can align our controlled distribution Q to Q^* as in equation (5.6), except we can now use our sampling distribution:

$$u_t^* = \mathbb{E}_{Q_s} [\omega(V) v_t], \quad (5.13)$$

with importance sampling weights:

$$\begin{aligned}
 \omega(V) &= \frac{1}{\eta} \exp \left(-\frac{1}{\lambda} \tilde{S}(V) \right) \left(\frac{p(V)}{q_s(V)} \right) \\
 &= \frac{1}{\eta} \exp \left(-\frac{1}{\lambda} \left(S(V) + \lambda \log \left(\frac{p(V)}{q_s(V)} \right) \right) \right) \\
 &\quad \times \exp \left(\log \left(\frac{p(V)}{q_s(V)} \right) \right) \\
 &= \frac{1}{\eta} \exp \left(-\frac{1}{\lambda} S(V) \right).
 \end{aligned} \quad (5.14)$$

Note that our change of cost equation (5.10) resulted in the optimal control being biased towards the sampling distribution, as shown in equation (5.12). However, this simplified the weights $\omega(V)$ and allowed us to design arbitrary sampling distributions Q_s . In [10], $S(V)$ was defined as the state-dependent cost. However, this restriction was made to relate the approach to path integral control [13]. Such relation was only shown exactly when P is the distribution induced by an uncontrolled continuous-time control-affine system. This restriction is not required in the Information-Theoretic framework, which allows for a larger class of systems, and one can add input costs in $S(V)$.

5.3.2 Sampling from Ancillary Controllers

There are several ways one could design an arbitrary sampling distribution. This paper focuses on taking most samples around a previously computed input distribution and some samples from hand-crafted policies.

In particular, we design a set of task-specific ancillary controllers, these being, e.g., open-loop motion primitives, reference tracking feedback controllers, or learning-based strategies to propose J input sequences $U^j = [u_0^j, u_1^j, \dots, u_t^j, \dots, u_{T_H}^j]$. These ancillary controllers are described for each experiment in Sections 5.4 and 5.5. We then choose the K sampled input sequences V_s^k as,

$$V_s^k = \begin{cases} U^j, & \text{with } j = k & \text{if } k \leq J \\ V^k, & \text{as in equation (5.9)} & \text{if } k > J, \end{cases} \quad (5.15)$$

meaning that, at each time step, we take one sample from each of the J ancillary controllers, and the remaining $K - J$ samples are taken according to the classical MPPI strategy.

5.3.3 Autotuning the Inverse Temperature

As in [85] and similarly to [83], we autotune the inverse temperature λ online based on the normalization factor η .

$$\lambda_{t+1} = \begin{cases} 0.9\lambda_t & \text{if } \eta > \eta_{max} \\ 1.2\lambda_t & \text{if } \eta < \eta_{min} \\ \lambda_t & \text{otherwise} \end{cases} \quad (5.16)$$

In all experiments, this can roughly keep the number of samples with a significant weight between η_{min} and η_{max} .

5.4 Illustrative Experiment

We apply our Biased-MPPI to a rotary inverted pendulum [97] (Figure 5.2) in simulation to visualize its main features.

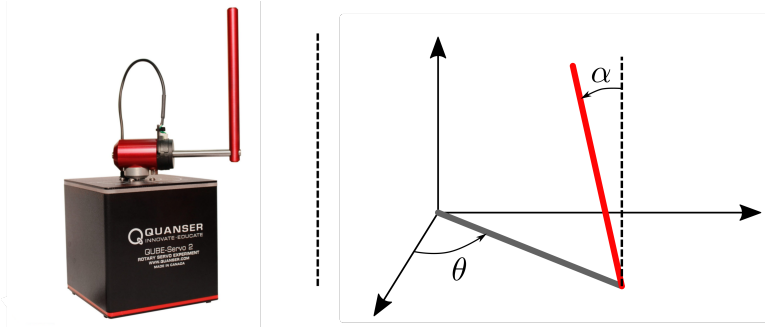


Figure 5.2: *Left*, Quanser Qube-Servo, and *right*, its diagram. The arm's rotation, θ , is the actuated angle. The angle between the pendulum and the upright position, α , is not actuated.

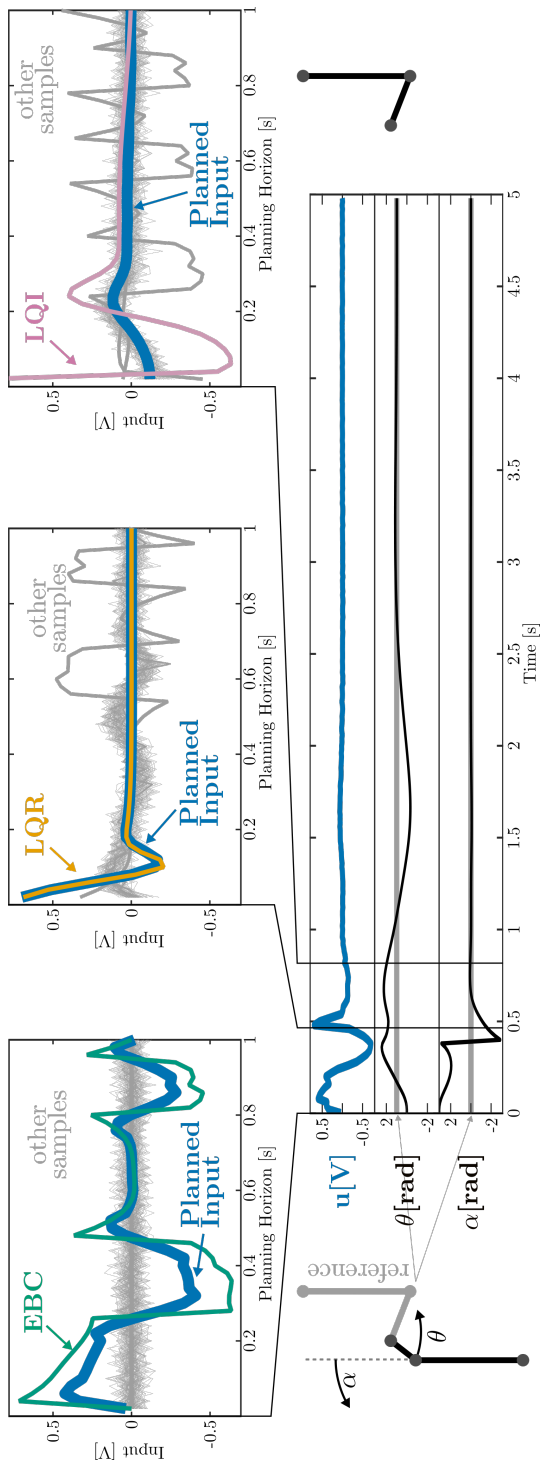


Figure 5.3: Input and state evolution during a pendulum experiment with Biased-MPPI. We show the samples taken and the resulting planned input sequence over the planning horizon for three instances. While we sample all ancillary controllers in each instance, we highlight the one with the most influence on the planned input sequence.

5.4.1 Swing-up and Tracking

Starting at the bottom equilibrium with $\theta_0 = 0$ and $\alpha_0 = \pi$, the task is to swing up the pendulum to $\alpha_r = 0$ while keeping the arm close to $\theta_r = 1$. Thus, the running cost is:

$$C_p(x(t)) = 100((\theta_t - \theta_r)^2 + (\alpha_t - \alpha_r)^2) + \dot{\theta}_t^2 + 2\dot{\alpha}_t^2. \quad (5.17)$$

The system has dynamics $x(t+1) = F(x(t), u(t))$, where the state of the system at time-step t is denoted as $x(t) = [\theta_t, \alpha_t, \dot{\theta}_t, \dot{\alpha}_t]^T$, and u represents the system's input. The nonlinear model is derived from the Lagrange equations. To design linear controllers, the model is linearized at the top equilibrium using Euler-Lagrange's method [98]. To showcase resilience against model uncertainties, the parameters of the simulation's pendulum model are multiplied by $1 + \gamma$ in each experiment, where $\gamma \sim \mathcal{N}(0, 0.05)$. The seed is consistent across methods. The system is discretized and controllers run at 50Hz, the controller plans $T_H = 50$ steps ahead (1s), covariance $\Sigma_s = 0.5$, $\eta_{min} = 2$ and $\eta_{max} = 5$.

Ancillary Controllers

We design three ancillary controllers as a baseline and to guide the sampling strategy.

A Linear Quadratic Regulator (LQR) designed using the `lqr` command in Matlab, stabilizes the pendulum at the top equilibrium.

A Linear Quadratic Integral (LQI) tracks the reference θ_r while maintaining the pendulum at the top equilibrium. It is synthesized with the `lqi` command in Matlab.

A nonlinear Energy-Based Controller (EBC) is designed as in [98] to swing up the pendulum to the top equilibrium by increasing the potential energy of the system [99].

Switching Controller

We introduce as baseline a switching strategy equation (5.18) that combines all ancillary controllers. It swings up the pendulum using the input from the EBC, u_{ebc} , until α is within $\alpha_{catch} = 0.2$ of the top equilibrium. The LQR controller, with u_{lqr} , then stabilizes the pendulum. Once the pendulum is close to the top equilibrium ($\alpha_{track} = 0.05$) with angular velocity below $\dot{\alpha}_{track} = 0.1$ rad/s, the LQI, with u_{lqi} , is engaged for reference tracking.

$$u = \begin{cases} u_{lqi}, & \text{if } (|\alpha| < \alpha_{track}) \cap (|\dot{\alpha}| < \dot{\alpha}_{track}) \\ u_{lqr}, & \text{if } (|\alpha| < \alpha_{catch}) \\ u_{ebc}, & \text{otherwise} \end{cases} \quad (5.18)$$

Results

Figure 5.3 depicts a pendulum experiment's input and state evolution with Biased-MPPI, also showcasing the samples taken and the ancillary controllers' influence on the plan. At the beginning of the experiment, ECB rapidly swings up the pendulum, heavily influencing Biased-MPPI's planned input. Once near equilibrium, LQR provides a stabilizing sequence, closely tracked by Biased-MPPI. As stability is achieved, LQI suggests an input sequence swiftly bringing the arm towards the reference, albeit with high velocities.

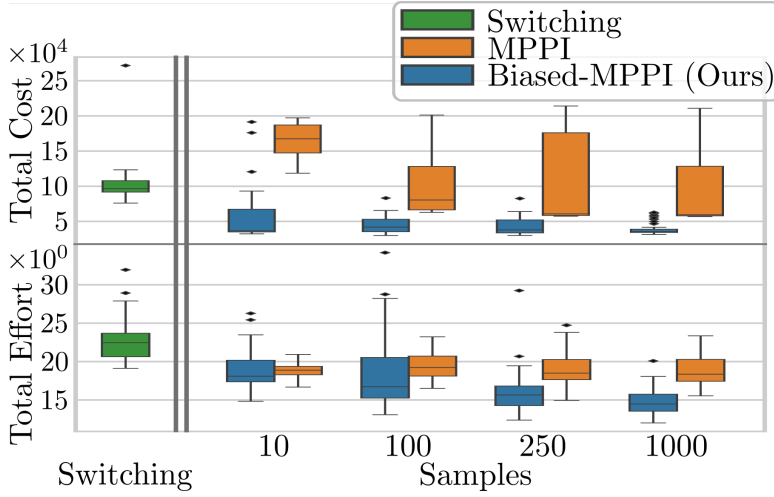


Figure 5.4: Total cost and control effort over 50 pendulum swing-ups with randomized model parameters.

Hence, Biased-MPPI, while influenced by LQI, opts for a lower amplitude input sequence due to cost function equation (5.17).

Figure 5.4 displays the distribution of total costs, defined as $\sum_{t=0}^{T_{end}} C_p(x(t))$ where $T_{end} = 250$ (5s) is the end of the episode, and the distribution of total efforts, defined as $\sum_{t=0}^{T_{end}} |u(t)|$, across 50 experiments. Biased-MPPI consistently outperforms both the switching strategy and the classic MPPI, regardless of the number of samples used. Moreover, the results indicate that including ancillary controllers in the proposed Biased-MPPI vastly improves the sampling efficiency, requiring fewer samples for better performance and enhancing the algorithm's robustness to model uncertainties.

5.5 Simulated Motion Planning Experiments

Interaction-Aware Model Predictive Path Integral (IA-MPPI) [28] is a decentralized communication-free motion planning method that predicts short-term goals of other agents with a constant velocity model and, under homogeneity and rationality assumptions, each agent simultaneously plans and predicts motions for all agents. In its cost function, IA-MPPI encourages adherence to navigation rules, such as giving the right-of-way to agents from the right and preferring the right-hand side during head-on encounters. We will investigate the effects of biasing its sampling scheme with ancillary controllers. The agents are vessels modeled using Roboat's model [52]. Controllers run at 10Hz, plan $T_H = 100$ steps ahead (10s), with $\Sigma_s = \text{diag}(6, 6, 0.12, 0.12)$, $\eta_{min} = 5$ and $\eta_{max} = 10$.

5.5.1 Solving an Intersection

An issue that can arise with classical MPPI formulation, which only takes samples around what was previously considered to be optimal, is the difficulty, once in one, of jumping out of local minima. This is particularly evident in IA-MPPI, especially in a crossing scenario.

In this experiment, depicted in Figure 5.5, two identical vessels start with zero velocity and have to cross each other's paths. In their cost function, described in previous work [28], the decentralized and communication-free IA-MPPI is encouraged to get each of the vessels across the intersection while being penalized for not yielding to the agent coming from the right-hand side.

Ancillary Controllers

To help switch out of local minima and improve sampling efficiency, four ancillary controllers are sampled using the proposed Biased-MPPI.

Go-Slow a sequence of inputs commanding a small amount of thrust to the vessel's side thrusters.

Go-Fast commands a large thrust.

Braking gives a zero velocity reference.

Go-to-Goal computes a velocity reference that takes each vessel towards its corresponding local goal at each time step of the planning horizon.

The velocity references proposed by the *Braking* and *Go-to-Goal* maneuvers are converted to input thrusts with a linear $\mathcal{H}_\infty$ controller, which is robust to model non-linearities, designed using the `musyn` command in Matlab.

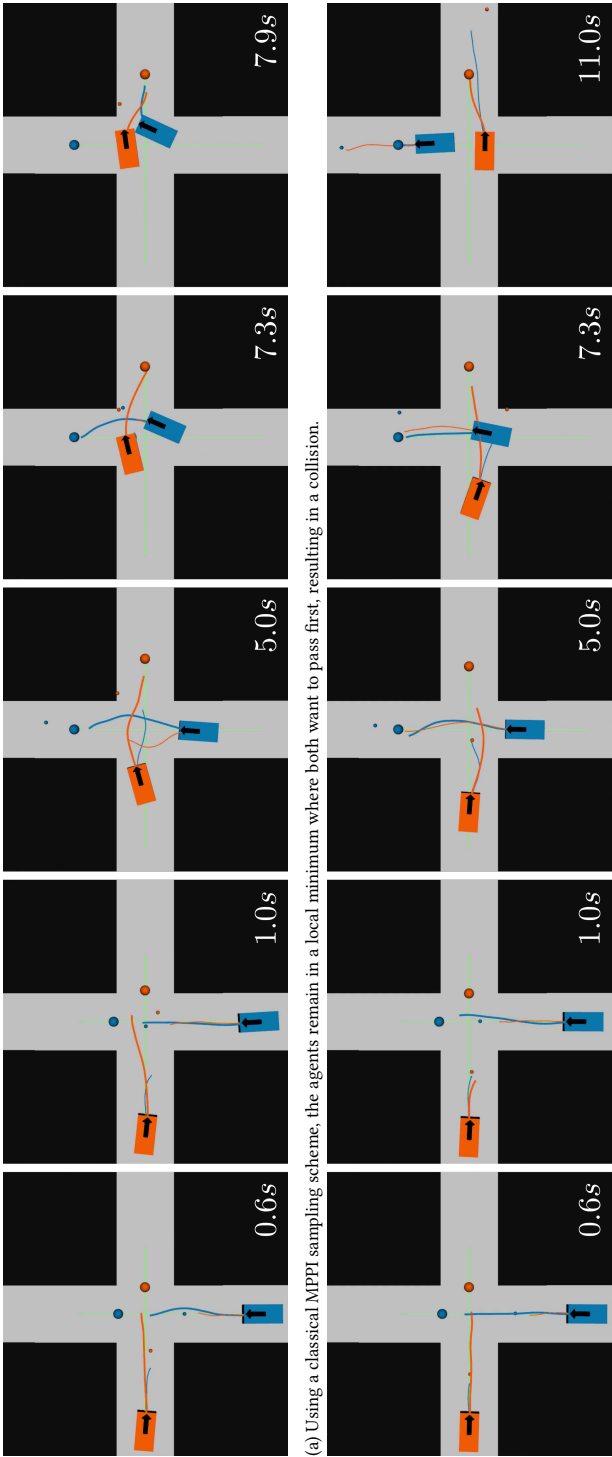
Results

With an initial velocity of zero, each agent anticipates an unobstructed intersection crossing. This expectation is based on a constant velocity prediction, as they assume the opposing agent will remain stationary. In Figure 5.5a, the classic IA-MPPI fails to switch from planning to cross first to a slower maneuver that yields since all of the samples are taken around the previous plan, leading to a collision. In Figure 5.5b, our Biased-IA-MPPI approach can swiftly switch between modes when it becomes evident that the vessel with the right-of-way will cross the intersection.

In Table 5.1, we see that in 50 experiments, our Biased-IA-MPPI achieves zero collisions and rule violations for any number of samples, compared to the IA-MPPI based on the classical MPPI sampling scheme, which results in several. Thanks to the ancillary controllers, our Biased-IA-MPPI also travels straight to the goal, reducing the distance traveled. While our Biased-IA-MPPI has a lower variance in arrival times, it is not always faster on average. This confirms the results proved in equation (5.12), i.e. the *Braking* and *Go-Slow* maneuvers are biasing towards a slower trajectory.

5.5.2 Interaction-Aware Planning with Four Vessels

To further test Biased-IA-MPPI, we run 50 experiments with randomized initial conditions and goals, where four agents have to navigate in cooperation in the Herengracht, an urban canal in Amsterdam, challenging due to its narrow sections under two bridges. The canal map and an example of successful navigation are shown in Figure 5.6.



(a) Using a classical MPPI sampling scheme, the agents remain in a local minimum where both want to pass first, resulting in a collision.

(b) Using the proposed Biased-IA-MPPI, the orange agent gives way to the blue agent as soon as it is clear that both agents want to cross.

Figure 5.5: Two vessels cross each other's path while penalized when not giving the right-of-way to agents coming from their right. The large circles are the agents' true local goals extracted from a global path. IA-MPPI is decentralized and communication-free, so the small dots are the goals vessels estimate of one another using constant velocity. The trajectories in blue are those the blue agent has planned for itself and predicted for the other, and the same goes for the orange agent.

Table 5.1: Results of 50 crossings for an increasing number of samples K . Metrics are reported for successful runs.

K Method	Collisions	Experiments With Rule Violations	Average Time to Arrival [s]	Average Distance Traveled [m]
50 IA-MPPI	4	9	16.41 $\pm$ 10.10	21.89 $\pm$ 8.433
50 Biased-IA-MPPI	0	0	17.64 $\pm$ 3.128	19.13 $\pm$ 2.466
200 IA-MPPI	10	4	12.77 $\pm$ 9.323	19.99 $\pm$ 10.16
200 Biased-IA-MPPI	0	0	12.66 $\pm$ 1.902	18.07 $\pm$ 2.012
500 IA-MPPI	7	11	11.02 $\pm$ 2.731	18.70 $\pm$ 3.518
500 Biased-IA-MPPI	0	0	11.43 $\pm$ 1.541	17.58 $\pm$ 1.880
1000 IA-MPPI	10	15	11.78 $\pm$ 3.823	19.31 $\pm$ 3.539
1000 Biased-IA-MPPI	0	0	11.00 $\pm$ 1.309	17.35 $\pm$ 1.625
2000 IA-MPPI	7	15	11.10 $\pm$ 4.101	19.72 $\pm$ 5.038
2000 Biased-IA-MPPI	0	0	10.68 $\pm$ 1.245	17.27 $\pm$ 1.716

Ancillary Controllers

We use all of the ancillary controllers described in Section 5.5.1. Additionally, we use a learning-based trajectory prediction model adapted and trained for urban vessels [57]. However, we do not use this model for predictions. We track the trajectories it provides with an $\mathcal{H}_\infty$ controller to generate input sequences, which Biased-MPPI can consider in its sampling scheme.

Results

In Table 5.2, results from 50 experiments show that with 50 samples, our Biased-IA-MPPI is cautious, leading to 10 deadlocks, possibly biased by the *Braking* maneuver. In contrast, the conventional IA-MPPI approach, without the ancillary controller, results in 16 collisions.

As the number of samples increases, the bias from the ancillary controllers diminishes, causing Biased-IA-MPPI to behave less conservatively. Consequently, the number of deadlocks approaches zero, but a few collisions may occur. With both methods, over half of the successful experiments incur at least a rule violation. In these crowded scenes, violations are common, e.g., not stopping to yield to an agent with priority when it is still relatively far away. Still, in both collision counts and the number of experiments resulting in rule violations, our Biased-IA-MPPI consistently outperforms IA-MPPI using the traditional sampling method.

Figure 5.7 displays both methods' quartiles, min, max, and outliers of successful experiments. The ancillary controllers direct the sampling distribution towards lower-cost areas of the state space, significantly reducing travel distances. Despite this, as predicted by equation (5.12), Biased-IA-MPPI also exhibits a bias towards slightly slower movement

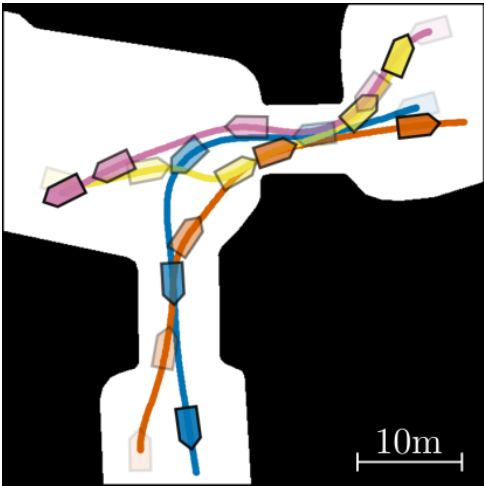


Figure 5.6: Four agents navigating in the Herengracht. Video.

Table 5.2: Results for 50 runs of four-agent experiments in the Herengracht with randomized initial conditions and goals for an increasing number of samples K .

K	Method	Successes	Deadlocks	Collisions	Experiments With Rule Violations
50	IA-MPPI	34	0	16	22
	Biased-IA-MPPI	40	10	0	18
200	IA-MPPI	43	1	6	34
	Biased-IA-MPPI	46	1	3	28
500	IA-MPPI	47	0	3	36
	Biased-IA-MPPI	49	0	1	35
1000	IA-MPPI	45	0	5	36
	Biased-IA-MPPI	50	0	0	33
2000	IA-MPPI	47	0	3	36
	Biased-IA-MPPI	49	0	1	34

due to “Braking” and “Go-Slow” maneuvers, resulting in similar travel times as the regular IA-MPPI.

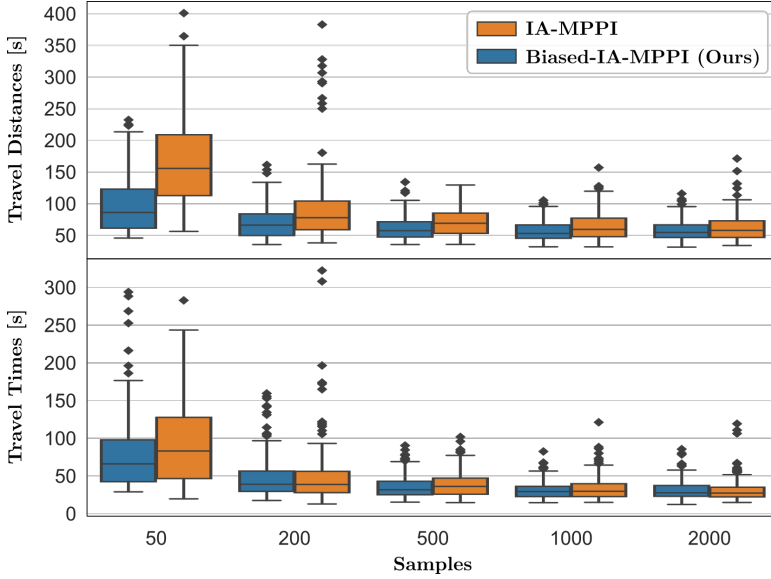


Figure 5.7: Agents’ traveled distance and travel time over 50 experiments in the Herengracht. Metrics are reported for experiments that were successful with both methods.

5.6 Real-World Motion Planning Experiment

A Clearpath Jackal robot attempts to drive to a goal as fast as possible ($\sim 2m/s$) while avoiding a box. Halfway through, the box is thrown in front of the robot. The position and the velocity of the box and the robot are estimated using information from a motion capture system. The velocity-controlled robot is modeled as a unicycle, and the box’s position is propagated through the planning horizon using a constant velocity model. The cost function is defined as,

$$C_j(x(t)) = \|p_{t,r} - p_g\| + 100(\|p_{t,r} - p_{t,b}\| < 0.5) \quad (5.19)$$

where $p_{t,r}$, p_g and $p_{t,b}$ are the position of the robot, the goal, and the box, respectively, at time t . Controllers run at 10Hz, plan $T_H = 50$ steps ahead (5s), with $K = 300$ samples, covariance $\Sigma_s = 0.5 \cdot I_{2 \times 2}$, $\eta_{min} = 5$ and $\eta_{max} = 10$.

Ancillary Controllers

We sample a *Braking* maneuver, i.e., a zero velocity reference throughout the horizon.

Results

Figure 5.8 shows the top 50 sampled trajectories sampled by (a), MPPI, and (b), our proposed Biased-MPPI. When the box is unexpectedly thrown in front of the robot, MPPI only samples trajectories that collide with the box. Given the cost function, MPPI prefers the samples that remain in collision for the least time. On the other hand, sampling also a zero velocity reference, Biased-MPPI quickly converges to a braking maneuver, avoiding the collision altogether. MPPI resulted in six collisions in over ten experiments, while Biased-MPPI resulted in none.

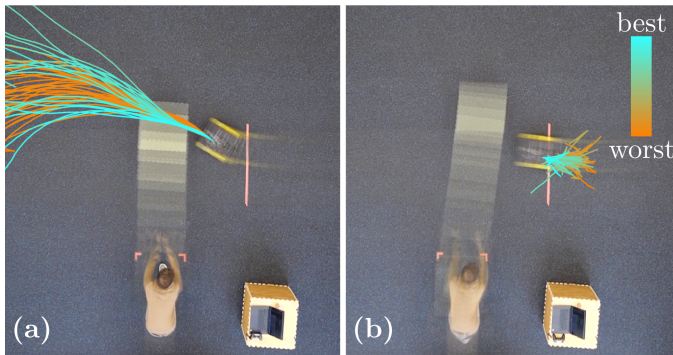


Figure 5.8: Visualized are the top 50 sampled trajectories, color-graded by their cost. (a) Classic MPPI is about to crash. (b) Our Biased-MPPI avoids collision. See video.

5.7 Conclusions

In this paper, we have derived a sampling scheme for MPPI control that removes computationally problematic terms and allows for the design of arbitrary sampling distributions as long as a bias in the solution is allowed. We proposed using classical and learning-based ancillary controllers for several control and motion planning experiments to bias the sampling distribution and achieve more efficient sampling and better performances. We demonstrated how the proposed algorithm can act as a control fusion scheme, taking suggestions from an arbitrary number of controllers and improving upon them. The resulting Biased-MPPI was shown to be better performing and more robust to model uncertainties compared to classical controllers and the baseline MPPI method, achieving faster swing-ups for a rotational inverted pendulum as well as safer, closer to optimal trajectories in interaction-aware motion planning experiments in constrained multi-agent environments, all while requiring less samples. The overall gains in safety, performance, and sample efficiency come at the expense of a potentially harmful bias, as shown with the sampling of *Braking* and *Go-Slow* maneuvers, which can result in slower trajectories. In the future, our approach could be employed as a potential solution to complex multi-modal problems. For example, a higher-level task planner could propose several ancillary controllers and alternative plans to be sampled to achieve global solutions.

6

Applications to Contact-Rich Robot Manipulation

In this chapter, we extend the work from previous chapters to a different domain, leveraging the gradient-free nature and parallelizable sampling of Model Predictive Path Integral control (MPPI) control. This characteristic makes MPPI particularly well-suited for use with IsaacGym, a GPU-based physics engine capable of parallelizing hundreds of simulations.

Creating a simulated version of the world in IsaacGym allows us to roll out hundreds of input sequences faster than in real-time to approximate optimal control. This approach is especially advantageous for contact-rich tasks like pushing or picking because of their difficult modeling and discontinuous nature. We utilize both prehensile and non-prehensile mobile manipulators and conduct experiments with real robots to demonstrate the effectiveness of this method.

Section 6.1 motivates applying MPPI with a parallelizable physics engine to contact rich tasks, reviews relevant literature, and states our contributions. Section 6.2 covers the necessary background and setup and proposes our new approach integrating MPPI with IsaacGym. We present several examples and experiments in Section 6.3, showcasing mobile manipulators in simulated and real-world environments in motion planning, pushing, and picking tasks. Finally, we conclude with a summary of findings and potential future directions in Section 6.4 and Section 6.5.

This chapter is a verbatim copy of the peer-reviewed paper [85]:

-  C. Pezzato*, C. Salmi*, **E. Trevisan***, M. Spahn, J. Alonso-Mora and C. H. Corbato, “Sampling-based Model Predictive Control Leveraging Parallelizable Physics Simulations,” *IEEE Robotics and Automation Letters (RA-L)*, January 2025. *Indicates equal contribution in alphabetical order.

Statement of contributions: Elia and Corrado contributed to the initial idea of using a physics simulator as a dynamic model for MPPI. Elia contributed to the theory of MPPI while Corrado contributed to the connection of MPPI with the physics simulator, designing and implementing the non-prehensile manipulation and whole-body control tasks. Chadi contributed to the software development of this method, and Max designed and performed the experiments for motion planning. Corrado, Elia, and Chadi performed the experiments on the real robot. Javier and Carlos provided discussions on the ideas and feedback, as well as editing of the manuscript.

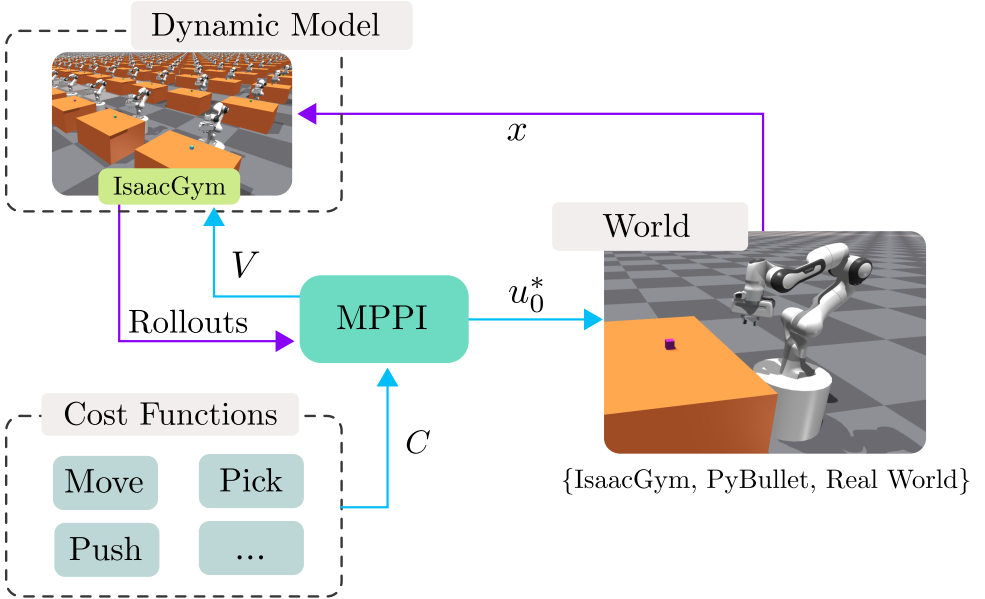


Figure 6.1: Scheme of the proposed method using IsaacGym as the dynamic model for MPPI. At each time step, IsaacGym is reset to the current world's state q , and random input sequences V are applied for the horizon T , to every environment. The resulting rolled-out trajectories are used by MPPI to approximate the optimal control u_0^* given a cost function C .

6.1 Introduction

As robots become increasingly integrated into our daily lives, their ability to navigate and interact with the environment is becoming more important than ever. From collision avoidance to moving obstacles out of the way to pick up some objects, robots must be able to plan their motions while accounting for contact with their surroundings. At the same time, robotic platforms are becoming more and more complex, and include many Degrees of Freedom (DOF) in order to achieve agile and dexterous movements. All this poses many challenging problems to motion planners, such as collision-free navigation in complex and dynamic environments, high DOF mobile manipulation, contact-rich tasks such as picking and pushing, and in-hand manipulation.

Solutions to these challenges exist but are often specialized and not easily transferable to different scenarios. Learning-based approaches, for example, can leverage physics simulators to train policies for complex tasks but require extensive training and resources. For instance, [100] took years to develop, utilizing 6144 CPU cores and 50 hours of training to learn a policy for in-hand cube manipulation. On the other hand, model-based approaches like Model Predictive Control (MPC) can solve challenging tasks [101]. However, MPC often relies on constrained optimization, requiring constraint simplifications, precise modeling, and ad-hoc solutions to handle discontinuous dynamics in contact-rich tasks [102, 103]. While utilizing motion memory for warm-starting optimization can enhance performance [104, 105], the above limitations still persist.

Recently, MPPI [51] and its information-theoretic counterpart [10] addressed optimal control problems via importance sampling, mitigating challenges tied to constrained optimization algorithms dealing with non-convex constraints and discontinuous dynamics. However, substantial modeling remains necessary. In this paper, we propose a training-free model-based framework for real-time control of complex systems, where one designs only a task cost function, not the problem’s dynamics and contact models. We introduce the idea of using a general GPU-parallelizable physics simulator, IsaacGym [106], as the dynamic model for MPPI. This creates a robust framework that generalizes to various tasks. An overview is given in Figure 6.1.

6.1.1 Related Work

This section provides an overview of selected works focusing on motion planning and contact-rich tasks in robotics. Motion planning pipelines are categorized as global and local motion planning [107]. Local motion planning encompasses approaches like operational space control, geometric methods such as Riemannian Motion Policies [108] and Optimization Fabrics [109, 110], and receding-horizon optimization formulations like MPC [111] that may incorporate learned components [112]. Most MPC algorithms rely on constrained optimization and assume smooth dynamics. However, contact-rich tasks pose challenges due to their non-smooth and hybrid nature, involving sticking and sliding frictions or entering contacts, requiring extensive modeling and ad-hoc solutions for pushing tasks [102, 103].

In contrast, MPPI control [10, 51] is a sampling-based MPC approach that approximates optimal control via parallel sampling of input sequences. MPPI is gradient-free and well-suited for systems with non-linear, non-convex, discontinuous dynamics and cost functions. It has successfully controlled high-degree-of-freedom manipulators in real-time [113], incorporating self-collision avoidance using trained neural networks and collision-checking functions [114]. However, these approaches have limited interaction with the environment. In [80], the authors propose ensemble MPPI, a variation that handles complex tasks and adapts to parameter uncertainty, but the task modeling remains unclear, and no open-source implementation is available.

To alleviate the problem of explicit modeling, some works have addressed the use of physics simulators for sampling-based MPC. In [115], the authors use the RaiSim simulator to sample waypoints for foot placement of a quadruped. Moreover, Howell et al. [84] proposed a sampling-based MPC method that employs the MuJoCo [116] as a dynamic model for rolling out sampled input sequences. This offloads modeling efforts to the physics engine, simplifying controller design. However, MuJoCo’s parallelization capabilities are constrained by the number of CPU threads, limiting real-time performance when many samples are required to solve a task. Moreover, results are presented only in simulation.

A high number of samples is particularly crucial in tasks such as non-prehensile manipulation with robot manipulators. Traditional approaches often involve sampling end-effector trajectories on a plane, relying on additional controllers for robot actuation and learned models for predictions [81, 82]. For instance, Arruda et al. [81] use a forward-learned model trained on 326 real robot pushes. This model is employed by an MPPI controller to

plan push manipulations as end-effector trajectories. Cong et al. [82] train a Long Short-Term Memory-based model to capture push dynamics using a dataset of 300 randomized objects. End-effector trajectories are sampled within a rectangular 2D workspace. Both methods require a separate controller to convert cartesian motions into joint commands, and both perform push manipulation through a sequence of pushes, resulting in discontinuous motion. These methods are not easily transferable to other robots, particularly for non-holonomic mobile pushing.

6.1.2 Contributions

This paper presents a novel open-source implementation of Model Predictive Path Integral (MPPI) control with a generic physics simulator as the dynamical model. This enables the method to solve many contact-rich motion planning problems. The two key contributions of this work are:

- The integration of the MPPI controller with the GPU-parallelizable simulator IsaacGym, distinguishing our approach from prior works in MPPI. Our method facilitates collision checking and contact-rich manipulation tasks leveraging the contact models and rigid body interactions included in the simulator without requiring gradients. Our solution allows smooth real-time control of real-world systems with high degrees of freedom, efficiently computing hundreds of rollouts in parallel.
- A versatile method applicable to various motion planning challenges, including collision avoidance, prehensile and non-prehensile manipulation, and whole-body control with diverse robots. We provide an open-source implementation that can be readily reused and extended to heterogeneous robots and tasks.

We perform many contact-rich tasks with several robotic platforms and real-world experiments. We include omnidirectional and differential drive robots and fixed or mobile manipulators and compare against many specialized baselines.

6.2 Sampling-based MPC via Parallelizable Physics Simulations

In this section we describe the integration of MPPI with IsaacGym, which enables real-time control of complex contact-rich robotic systems with minimal modeling.

6.2.1 Background Theory on MPPI

In this section, we give an overview of the background theory of MPPI. For more theoretical insights, please refer to the original publications [10, 51]. MPPI is a method to solve stochastic optimal control problems for discrete-time dynamical systems such as

$$x_{t+1} = f(x_t, v_t), \quad v_t \sim \mathcal{N}(u_t, \Sigma), \quad (6.1)$$

where the nonlinear state-transition function f describes how the state x evolves over time t with a control input v_t . MPPI samples K noisy input sequences V_k . These sequences are then applied to the system to simulate K state trajectories Q_k , $k \in [1, K]$, over a time

horizon T :

$$Q_k = [x_0, f(x_0, v_0), \dots, f(x_{T-1}, v_{T-1})]. \quad (6.2)$$

Given the state trajectories Q_k and a designed cost function C to be minimized, the total state-cost S_k of an input sequence V_k is computed by functional composition $S_k = C(Q_k)$. Then, each rollout is weighted by importance sampling weights w_k , computed via an inverse exponential of S_k with tuning parameter β , normalized by η . The minimum sampled cost $\rho = \min_k S_k$ is subtracted for numerical stability, leading to:

$$w_k = \frac{1}{\eta} \exp\left(-\frac{1}{\beta}(S_k - \rho)\right), \quad \sum_{k=1}^K w_k = 1 \quad (6.3)$$

The parameter β is also known as *inverse temperature*. The weights are then used to compute the approximate optimal control input sequence U^* :

$$U^* = \sum_{k=1}^K w_k V_k \quad (6.4)$$

Equation (6.3) and Equation (6.4) demonstrate the approach taken to approximate the optimal control. Equation (6.4) represents a weighted average of sampled control inputs, while Equation (6.3) assigns exponentially higher weights to less costly inputs. The first input u_0^* of the sequence U^* is applied to the system. Then the process is repeated.

6.2.2 Proposed Algorithm

We now describe how we use MPPI with IsaacGym, summarized in Algorithm 2. We initialize an input sequence U_{init} as a vector of zeroes with a length of T , where T is the time horizon in steps. We then sample K sequences of additive input noise $\mathcal{E}_k$ for exploring the input space around U_{init} . The key concept is that, instead of explicitly defining a nonlinear transition function f , we use IsaacGym to compute the next state x_{t+1} given x_t and control input v_t . This is done by reading the current state of the environment, resetting the state of the simulator to the observed values, and then applying the noisy control input sequence to simulate the state trajectories in IsaacGym. Note that these K state trajectories can be computed independently of each other. We make use of this property to forward simulate all the rollouts in parallel leveraging the parallelization capabilities of IsaacGym. Instead of sampling from a Gaussian distribution, we follow the strategy of a recent paper [113] that proposes to sample *Halton Splines* instead for better exploration and smoother trajectories. Similar to [113], we fit B-Splines to inputs sampled from a Halton sequence using standard Python modules and then we evaluate the spline at regular intervals to retrieve $\mathcal{E}_k$. Unlike [113], we do not update the variance of the sampling distribution. Instead, we keep it as a tuning parameter, constant during execution. Updating the variance as in [113] can lead to better convergence to a goal, but it also leads to stagnation of the control over time, which is harmful in the contact-rich tasks considered in this paper. Once the task begins, we reset our K simulation environments on IsaacGym to the current observed *world* state x . In parallel, we can now roll out the sampled input sequences V_k into

state trajectories Q_k using K simulation environments on IsaacGym and compute their corresponding cost S_k using the designed *cost function* C . The cost is discounted over the planning horizon T by a factor γ [113]:

$$S_k = \sum_{t=0}^{T-1} \gamma^t C(x_{t,k}, v_{t,k}) \quad (6.5)$$

Next, we can compute the *importance sampling* weights w_k as in Equation (6.3). Note that the normalization factor η is a useful metric to monitor, as it indicates the number of samples assigned significant weights. We use this to tune β for the next iteration such that η is maintained within an upper and lower bound:

$$\beta_{t+1} = \begin{cases} 0.9\beta & \text{if } \eta > \eta_{max} \\ 1.2\beta & \text{if } \eta < \eta_{min} \\ \beta & \text{otherwise} \end{cases} \quad (6.6)$$

Empirically, we observed in all performed tasks that setting $5 < \eta < 10$ is a good balance for smooth behavior. Finally, an approximation of the optimal control sequence U^* can now be computed via a weighted average of the sampled inputs Equation (6.4). U_{init} is now updated with U^* , time-shifted backward of one timestep so that it can be used as a warm-start for the next iteration, $U_{init} = [u_1^*, \dots, u_{T-1}^*, u_{T-1}^*] \in \mathbb{R}^T$. The second last input in the shifted sequence is propagated to the last input as well. From the sequence U^* , only the first input u_0^* is applied to the system, and the next iteration starts.

Algorithm 2 Proposed Approach

```

1: Initialize:
    $U_{init} = [0, \dots, 0]$  ▷  $U_{init} \in \mathbb{R}^T$ 
    $\mathcal{E}_k \leftarrow \text{sampleHaltonSplines}()$  ▷  $k = 1 \dots K$ 
2: while taskNotDone do
3:    $x \leftarrow \text{observeEnvironment}()$ 
4:    $\text{resetSimulations}(x)$ 
5:   for  $k = 1 \dots K$  do ▷ in parallel
6:      $V_k = U_{init} + \mathcal{E}_k$ 
7:      $[Q_k, S_k] \leftarrow \text{computeRolloutCost}(V_k, \gamma)$ 
8:      $w_k \leftarrow \text{importanceSampling}$  ▷ equation (6.3)
9:   end for
10:   $\beta \leftarrow \text{updateBeta}(\beta, \eta)$  ▷ equation (6.6)
11:   $U^* = \sum_{k=1}^K w_k V_k$ 
12:   $U_{init} \leftarrow \text{timeShift}(U^*)$ 
13:   $\text{applyInput}(u_0^*)$ 
14: end while

```

6.2.3 Exploiting the Physics Simulator Features

IsaacGym provides useful information and general models that are particularly useful for robot control in contact-rich tasks. Besides being useful to simulate the physical interaction of rigid bodies, we leverage IsaacGym for *collision checking* and *tackling model uncertainty* with domain randomization.

Collision checking

Collision checking in robotics can be challenging for a number of reasons, one of them being computational complexity. This is particularly true if the task requires continuous collision checking as the robot moves in dense environments with complex object shapes. To overcome this problem, approximations are often introduced with the convexification of the space. However, this requires several heuristics and can hinder robot motions in complex scenes.

Instead, we propose to tackle the problem of collision checking by using the already available *contact forces tensor* from IsaacGym, which is available for each simulation step. To avoid collisions, we then define a cost function proportional to the contact forces for the MPPI:

$$C_{coll} = \omega_c \sum F_{obst}, \quad (6.7)$$

where F_{obst} are the contact forces exerted on the different obstacles. This allows us to perform continuous collision checking at each time step over the horizon T , with arbitrary complex shapes. By heavily penalizing contacts with obstacles, the robot will avoid collisions. On the other hand, by relaxing the weight ω_c one can allow for certain contacts required for the task, such as rolling a ball against a wall (Section 6.3.3).

Tackling model uncertainty

IsaacGym is designed to easily support domain randomization. We use this feature to randomize the object properties in each environment in case of contact-rich tasks, such that uncertainty is incorporated in every rollout for the MPPI. Effectively, this allows to account for uncertainty in environment perception. Specifically, starting from nominal physics properties, in every rollout objects are spawned with uncertainty on mass and friction nominal values, sampled from a uniform distribution. Additionally, the object size is also randomized with additive Gaussian noise, see Section 6.3 for experiment-specific details. Therefore, every simulation is different from the others, and all simulations are different from the *world* such that we can account for model mismatch. In a sense, we perform a sort of domain randomization in real-time to address the challenge of model uncertainty and imperfect perception. In the next section, we evaluate the performance of our method in different scenarios.

6.3 Experiments

We perform several experiments in three different categories: 1) *motion planning and collision avoidance*, 2) *whole-body control* of highDOF systems in contact-rich settings, and 3) *non-prehensile manipulation*. Experiments and simulations are conducted on an Alienware Laptop with Nvidia 3070 Ti graphics card. The software implementation consists of our open-source Python package that can easily be installed, tested, and extended to new

robots and tasks. In real-world tests, we used a Robot Operating System (ROS) wrapper to connect the robot to the planner and a motion capture system to determine the pose of manipulated objects. Our implementation allows for position, velocity, and torque control. In this paper, all robots are velocity-controlled except for the mobile manipulator in Section 6.3.2, which is torque-controlled.

6.3.1 Motion Planning and Collision Avoidance

We compare the performance of the proposed method in a pure local motion planning setting, i.e. no interaction with the environment. This aims to showcase the fact that our method is comparable to state-of-the-art techniques when no contact is involved. The main focus is the quantitative analysis of the method compared to two baselines, specifically optimization fabrics as presented in [109, 110] and a simple MPC formulation solved with ForcesPro [117]. We make use of an already available benchmark setup, the *localPlannerBench* [118]. We present results for two cases, namely a holonomic robot, and a robotic arm (Franka Emika Panda). For all experiments, we randomize five obstacles and the goal positions in $N = 100$ runs, see Figure 6.2 for some examples. Solutions by the three methods are assessed using four metrics, e.g. time to reach the goal, path length, solver time, and minimum clearance.

The compared methods show minimal differences in path length clearance for both examples (Figure 6.3). However, our method consistently reaches the goal faster (Figures 6.3 and 6.4, and Table 6.1). This is attributed to the perfect representation of the robot’s collision shapes used in our method, compared to the enclosing spheres in the ForcesPro MPC and optimization fabrics. It should be noted that our approach incurs higher computational times (Table 6.1) due to the physics simulations performed by IsaacGym. Despite this, our method remains competitive in motion planning applications and offers significant advantages in contact-rich tasks, as demonstrated in the following sections.

6.3.2 Prehensile Manipulation with Whole-Body Control

Our approach scales well with the complexity of the robot. In Figure 6.5, the task is to relocate an object from a table to an $[x, y, z]$ location using a mobile manipulator with 12DOF.

Although this is arguably a complex task for a robot, which usually requires manual engineering of a sequence of movements, such as navigation to a specific base goal, and

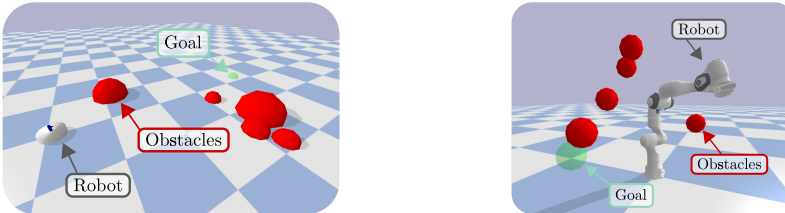


Figure 6.2: Examples for pure motion planning benchmark setups. Left: point robot with 3DOF. Right: manipulator with 7DOF.

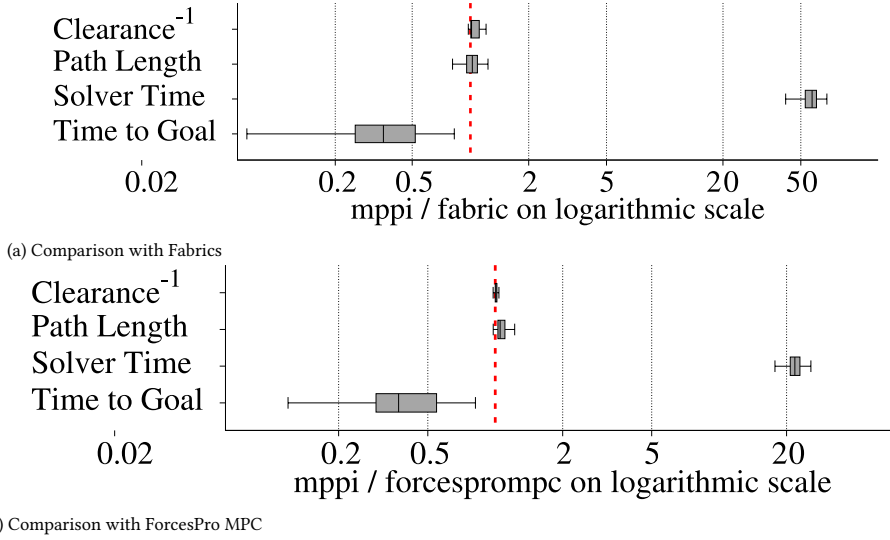


Figure 6.3: Results in pure motion planning problems for point robot with 3 DOF.

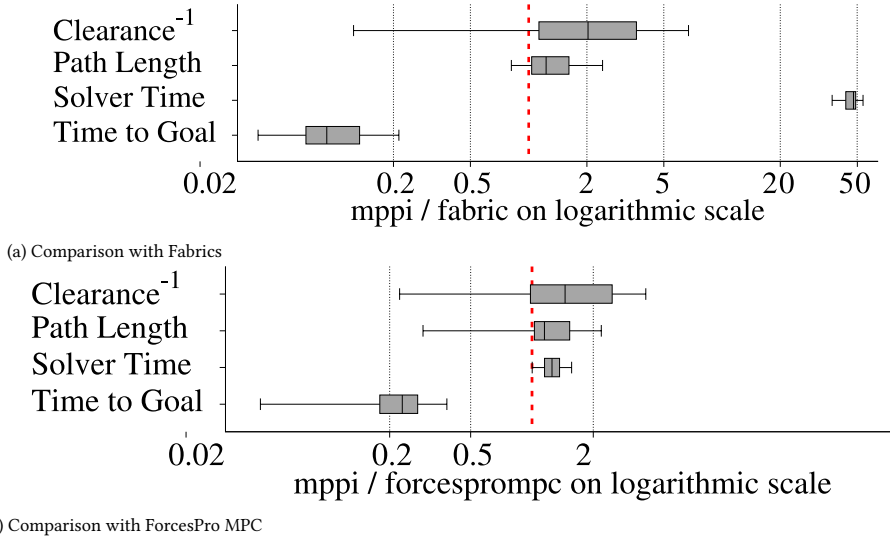


Figure 6.4: Results in pure motion planning problems for a robot manipulator with 7 DOF.

pre-post grasps, the solution is rather simple with our method. In fact, we specify the following cost function for the task:

$$C_{pick} = C_{dist} + C_{pose} + C_{coll} + C_{vel}, \quad (6.8)$$

where we consider the Euclidean distance of the end-effector to the object and the ob-

Table 6.1: Summary of motion planning experiments

Metric	Robot	Fabric	ForcesPro MPC	MPPI
Average Solver Time	Point	1.0ms	2.5ms	55ms
	Panda	1.4ms	51ms	63ms
Average Time to Goal	Point	7.4s	6.1s	2.7s
	Panda	9.6s	4.2s	0.8s

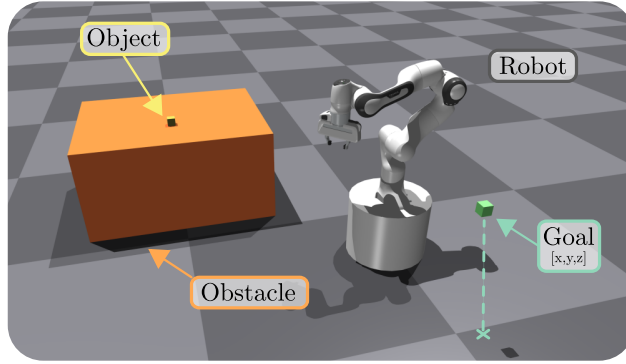


Figure 6.5: Whole-body motion of a mobile manipulator moving a cube from initial to a desired location.

ject to the goal: $C_{dist} = \omega_t \|p_{EE} - p_O\| + \omega_O \|p_G - p_O\|$. We give an incentive to keep the robot in a comfortable pose by penalizing deviations from a desired arm and gripper pose, end-effector orientation, as well as imposing a minimum end-effector height $C_{pose} = C_{Parm} + C_{Pgrip} + C_{Oee} + C_{Hee}$. We minimize collisions penalizing the forces on the table C_{coll} . Lastly, we penalize high arm and base velocities $C_{vel} = C_{Varm} + C_{Vbase}$ since, in this experiment, we torque-control the robot. By sampling all theDOF at once, including the base and the gripper, we achieve a fluid motion from start to end with no added heuristics for pick positions. We performed ten pick-and-deliver tasks, and the time taken was $15.67 \pm 7.21s$. The high standard deviation is because sometimes the cube falls, but the robot can recover by picking it up again from the floor.

For smooth whole-body motions of highDOF systems like this, many samples are required. Empirically, when the number of samples exceeds 50, a GPU pipeline is computationally cheaper than a CPU and scales better. Using IsaacGym, we can compute all the 750 samples required for mobile manipulation in parallel, computing the next control input online at 25Hz.

6.3.3 Non-Prehensile Manipulation

One advantage of using a physics simulator is that one can leverage generic physics rules for contacts, thus eliminating the need for learning or engineering specialized contact

models. We demonstrate this in non-prehensile manipulation tasks involving a 7-DOF arm (Figure 6.6) and two different mobile robots (Figures 6.7, 6.8 and 6.9). In Section 6.3.3, we apply our method to the two pushing tasks tackled in [81, 82] and we compare with their final results. Additionally, in Section 6.3.3, we demonstrate the ease of transferring our approach to different robots, including differential-drive.

Comparison with baselines for pushing with a robot arm

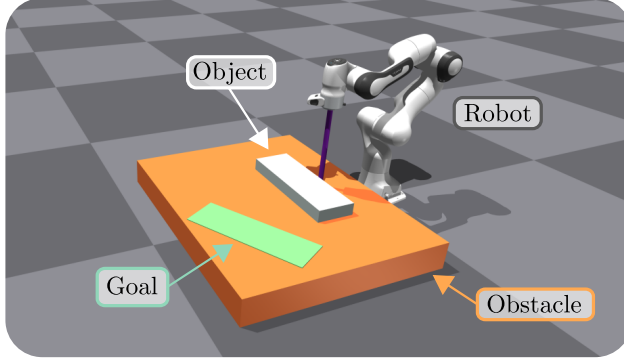


Figure 6.6: Example of non-prehensile push task with a 7-DOF robot arm using an object from [82].

6

We consider two baselines for non-prehensile pushing. In the first one, [81] tackles the problem of pushing a relatively small object to a target pose with either 0 (Pose 1) or 90 deg (Pose 2). They also consider sequences of push actions starting far from the object. In the second baseline, [82] considers 5 relatively big objects and assumes the robot's end effector is close to the object during execution. Since we do not have access to the same hardware, and the authors of the considered baselines do not provide their models and data, we only compare against their final results. Whenever their data cannot be retrieved for a certain result, we indicate this as *n/a*. We set up our simulation to match as close as possible the tasks in the baseline using the available information from the papers. Finally, we tune our method for the two tasks separately for a fair comparison with the individual baselines. The approach in [81] utilizes an MPPI in combination with a learned model for predicting pushing effects on an object. The authors sample 2D end-effector trajectories and then rely on inverse kinematic solvers, achieving push manipulation as a sequence of disconnected pushes. In contrast, we use MPPI to sample the control input directly as joint velocities in IsaacGym. By doing so, we achieve smooth continuous pushes where end-effector repositioning emerges naturally, and learning is not required. The cost function to be minimized for the task is:

$$C_{push} = C_{dist} + C_{push\ align} + C_{ee\ align}. \quad (6.9)$$

C_{dist} has the weighted distance *robot-object*, and *object-goal*:

$$C_{dist} = \omega_t \|p_R - p_O\| + \omega_{O_p} \|p_G - p_O\| + \omega_{O_r} \|\psi_O - \psi_G\|,$$

where p_G and ψ_G are the goal's position and orientation, while p_R and p_O denote the end-effector tip and block positions, respectively. The cost function $C_{push\ align}$ promotes

keeping the object between the robot and the goal. It is computed as $\cos(\alpha) + 1$, where α is the angle between the *robot-object* ($p_R - p_O$) and *goal-object* ($p_G - p_O$) vectors:

$$C_{push\ align} = \omega_a \left(\underbrace{\frac{(p_R - p_O) \cdot (p_G - p_O)}{\|p_R - p_O\| \|p_G - p_O\|}}_{\cos(\alpha)} + 1 \right). \quad (6.10)$$

We promote the end-effector to maintain a downward orientation at height d_h using pitch θ and roll ϕ :

$$C_{ee\ align} = \omega_{ee_r} \|[\phi, \theta] - [0, 0]\| + \omega_{ee_h} \|p_{R_z} - d_h\|.$$

The cost is minimized when the end-effector is close to the block at a certain height and orientation, and the block is between the end-effector and the goal at the desired goal pose¹.

We perform the same task as in [81] and compare the final results of pushing a squared object on a table surface to two poses (Pose 1 and 2) with a robot arm equipped with a stick. In Table 6.2, we report our findings, with our method showing double the accuracy. Our approach performs continuous pushes, unlike the baseline that stops for replanning after each short push. Thus, we complete either task in approximately 8 seconds, while the baseline takes approximately 4 minutes. We used the same evaluation metric of [81] for the final cost that is a weighted average of position and orientation errors: $1.5(|p_{G_x} - p_{O_x}| + |p_{G_y} - p_{O_y}|) + 0.01|\psi_O - \psi_G|$. For every run, the object is also randomized in the same way as the rollouts. See the accompanying video for the actual behavior.

Table 6.2: Summary of comparison with [81]

Approach	Start pose	Final cost
Baseline [81]	Pose 1	0.057
	Pose 2	0.079
Ours (sim)	Pose 1	0.029 $\pm$ 0.09
	Pose 2	0.03 $\pm$ 0.12

We further compare our approach with [82] in terms of the success rate of non-prehensile manipulation. Particularly, we consider the same task settings, that is pushing 5 different objects to 3 different goal poses. To do so we simply change the objects in the simulation and slightly re-tune the MPPI².

Since the trained models from [82] are not provided, we only compare the final results, reported in Table 6.3. Again, thanks to the continuous pushes, our method takes about 3 seconds per task, while the baseline needs about 24 seconds. We performed 10 pushes per object, totaling 150 pushes. For the non-prehensile manipulation task with the robot arm, the mass and friction of manipulated objects have 30% uncertainty, and table friction

¹Tuning: $\omega_t = 1$, $\omega_{O_p} = 16$, $\omega_{O_r} = 2$, $\omega_{ee_h} = 8$, $\omega_{ee_r} = 0.5$, $\omega_a = 0.8$, $dt = 0.04$, $T = 8$, $K = 500$.

²Tuning: $\omega_t = 5$, $\omega_{O_p} = 25$, $\omega_{O_r} = 21$, $\omega_{ee_h} = 30$, $\omega_{ee_r} = 0.3$, $\omega_a = 45$, $dt = 0.04$, $T = 8$, $K = 500$.

has 90% uncertainty on the nominal value, sampled uniformly. Size is randomized with zero-mean additive Gaussian noise with a 2 mm standard deviation.

Table 6.3: Summary of comparison with [82]

Approach	Metric	Object				
		A	B	C	D	E
Baseline [82]	Success [%]	93.5	90.9	93.9	91.6	89.5
Ours (sim)	Success [%]	100	93.3	96.7	100	66.7

Our method outperforms both baselines in terms of time to completion, accuracy, and success rate, except for one manipulated object. We achieve this without limiting the sampling to 2D end-effector trajectories, without needing learned models, and without requiring inverse kinematics solvers.

Extension to different robots

Our method is also easily extensible to different robot platforms and objects because it does not require specialized models or controllers that are robot specific, as opposed to the baselines considered. We chose to use an omnidirectional base, and a differential drive robot, to push a box or a sphere to a goal from different initial configurations. To do so, we only need to change the environment and robot Unified Robot Description Format (URDF) in IsaacGym, and re-tune the cost function for pushing due to different hardware.

6

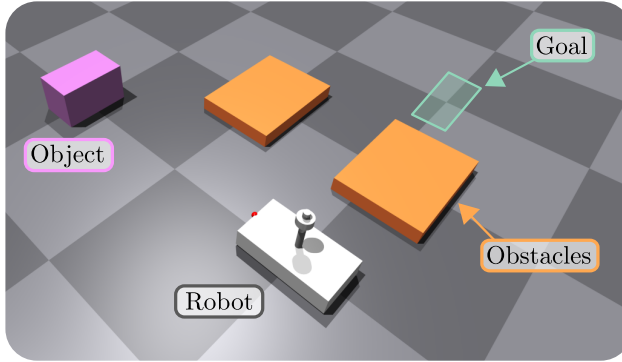


Figure 6.7: Non-prehensile task using an omnidirectional base. $\omega_t = 0.2$, $\omega_{O_p} = 2$, $\omega_{O_r} = 3$, $\omega_a = 0.6$, $\omega_c = 10$, $T = 8$, $dt = 0.04$, $K = 300$.

Omnidirectional push of a box The first task is the non-prehensile pushing of a box with an omnidirectional base, see Figure 6.7. Success is defined when the box is placed at the goal within 5cm in the $x - y$ direction and within 0.17 radians in rotation. The robot cannot touch obstacles. The cost function for the MPPI is the same as in Equation (6.9), re-tuned without considering end effector height and orientation since we now operate

on a plane. We add an explicit term for collision avoidance $C_{coll} = \omega_c \sum F_{obst}$:

$$C_{push} = C_{dist} + C_{push\ align} + C_{coll}, \quad (6.11)$$

Omnidirectional push of a sphere One can easily extend the example above to different objects with very different dynamics. We chose a sphere instead of a box, and we simply change the object spawned in the simulation. For this task, we want to put the ball in between the two walls, Figure 6.8. We considered multiple runs from two different starting poses, A and B. Results are summarized in Table 6.4 and the execution can be seen in the accompanying video.

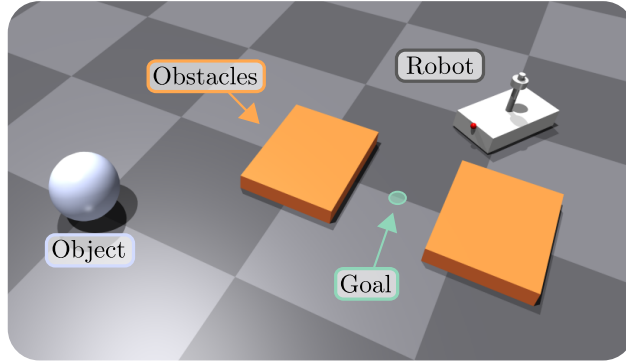


Figure 6.8: Rolling ball non-prehensile pushing. Goal: Ball placement between two obstacles. $\omega_t = 0.2$, $\omega_{O_p} = 0.1$, $\omega_{O_r} = 0$, $\omega_a = 0.1$, $\omega_c = 0.001$, $T = 8$, $dt = 0.04$, $K = 300$.

Table 6.4: Results with omnidirectional base

Obj	Env.	Runs	Time [s]
Box	Pose A	5	9.66 ± 0.84
	Pose B	5	12.84 ± 0.564
Sphere	Pose A	5	8.76 ± 0.38
	Pose B	5	7.45 ± 0.59

Differential drive non-prehensile pushing We perform differential drive non-prehensile pushing, see Figure 6.9, with the same cost function as before (see Equation (6.11)) but re-tuned. One can indeed change the robot for the task by simply changing the URDF, neglecting all the additional contact modeling that would be required in a classical model-based MPC. The time taken to push the box to the goal was 18.31s. In the mobile non-prehensile pushing experiments, objects to manipulate are spawned with 30% uncertainty on mass and friction sampled uniformly, while object size is randomized with Gaussian noise with a standard deviation of 5mm.

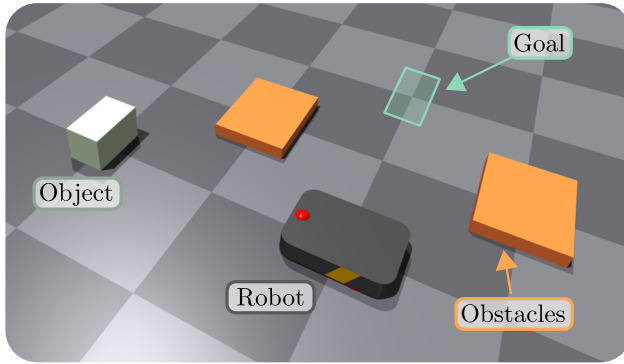


Figure 6.9: Non-prehensile differential drive pushing. Same task as omnidirectional base. $\omega_t = 0.1$, $\omega_{O_p} = 2$, $\omega_{O_r} = 3$, $\omega_a = 0.6$, $\omega_c = 100$, $T = 12$, $dt = 0.04$, $K = 400$.

6.3.4 Real-World Experiments

To demonstrate the applicability of our approach, we transfer to the real world a subset of the non-prehensile manipulation tasks previously presented in Section 6.3.3 with both the robot manipulator and the omnidirectional base. In particular, in Figure 6.10, we show the results of the 7 DOF manipulator pushing a product to two different goals, similar to the simulations corresponding to Table 6.2. As presented in Figure 6.1, the samples are rolled out in $K = 500$ simulated environments in IsaacGym which, at each timestep, are initialized to the state of the real world. Based on this, the optimal control is estimated and applied to the real system.

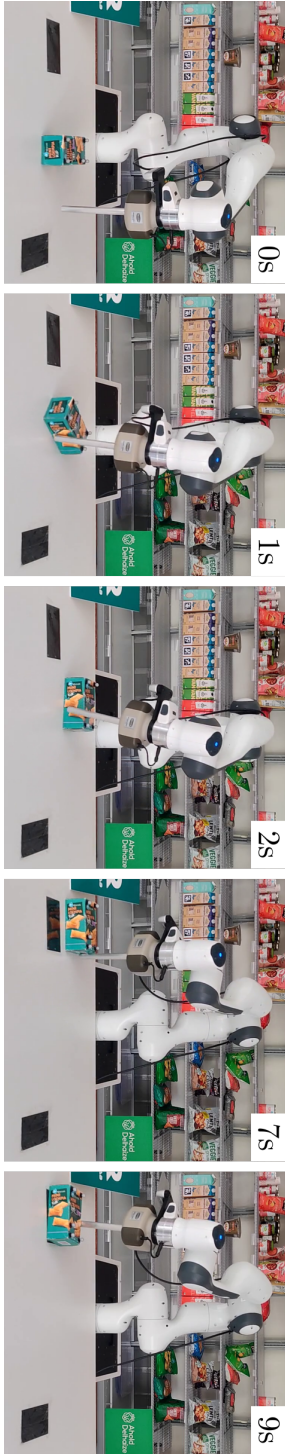
When transferring to the real world, compared to the experiments in Section 6.3.3, only the weights of the cost function were re-tuned. The horizon, control frequency, number of samples, structure of the cost function, and randomization of the sampled environments remained unchanged.

From the experimental evaluation on the real robot, we observe that the time to complete the different pushing tasks and the final position error are comparable to the results in the simulation from Table 6.2. Importantly, these results are achieved without taking assumptions on specific contact points, thus the robot can naturally re-position itself and change contact location autonomously. Additionally, our method allows to sample joint velocities directly thus we do not restrict the sampling to 2D end-effector trajectories to be then translated into joint commands as often seen in other approaches.

Lastly, to demonstrate robustness, we disturb by hand the execution of pushing tasks with the manipulator and the omnidirectional base. Since we do not assume the robot to be behind the object to be pushed for successful execution, and since the planning and execution happen in real-time at $25Hz$, we can largely perturb the task and let the robot compensate.



(a) Pushing straight to the goal on the right.



(b) Pushing to the goal on the left with 90° rotation.

Figure 6.10: Pushing to two goals with a 7 DOF manipulator directly controlling all joint velocities. Our method allows the end-effector to re-position around the object without specifying any desired contact point.



Figure 6.11: Qualitative real-world experiments with disturbances. The behavior can be seen in the accompanying video.

6.4 Discussion

In this section, we discuss key aspects and potential future work related to our solution. First, the computational demands of planning and control with our method can be high when extending the time horizon to several seconds. To keep the time horizon limited for real-time control while preventing being trapped in local minima, future work should incorporate global planning techniques such as A*, Rapidly exploring Random Tree (RRT), and Probabilistic Roadmaps (PRMs) [119] to guide the local planner. Similarly to warm starting predictive controllers [104, 105], one could make use of motion libraries of previous executions or learned policies along with random rollouts, to improve the sampling efficiency and exploration [71]. Second, in real-world scenarios, uncertainties and discrepancies between simulated and actual environments could present challenges for achieving precise movements and manipulation. We utilized randomization of object properties in the rollouts to address some uncertainties. However, online system identification to converge to the true model parameters is not performed. Enhancing the robustness of the MPPI algorithm itself by reducing model uncertainty, as demonstrated by [80], could further improve performance. Third, tuning control algorithms for optimal performance is time-consuming. Implementing autotuning techniques can automate the process and reduce manual effort. Finally, incorporating additional sensor support, such as lidars and signed distance fields, could be beneficial.

6

6.5 Conclusions

We presented a way to perform MPPI that uses a physics simulator as the dynamic model. By leveraging the GPU-parallelizable IsaacGym simulator for parallel sampling of forward trajectories, we have eliminated the need for explicit encoding of robot dynamics, contacts, and rigid-body interactions for MPPI. This makes our method easily adaptable to different objects and robots for a wide range of contact-rich motion-planning tasks. Through a series of simulations and real-world experiments, we have demonstrated the effectiveness of this approach in various scenarios, including motion planning with collision avoidance, non-prehensile manipulation, and whole-body control. We showed how our method can compete with state-of-the-art motion planners in case of no interactions, and how it outperforms by a margin other approaches for contact-rich tasks. In addition, we provided an open-source implementation that can be used to reproduce the presented results, and that can be adapted to new tasks and robots.

7

Conclusion and Discussion

The concluding chapter of this thesis reviews and analyzes the key findings. We examine the benefits and limitations of Model Predictive Path Integral control (MPPI), comparing it with optimization-based Model Predictive Control (MPC). Additionally, we highlight potential research directions that could further enhance this type of motion planner. The chapter concludes with the author's perspective on the future of Autonomous Surface Vessels (ASVs).

7.1 Conclusion

This thesis advances the motion planning of autonomous surface vessels, focusing on their navigation through crowded urban canals alongside human-driven vessels. Specifically, we concentrated on a sampling-based MPC method known as MPPI due to its computational efficiency and gradient-free nature. We extended MPPI to simultaneously perform prediction and planning for local motion planning in decentralized, communication-free, multi-agent navigation tasks. Additionally, we explored how sampling ancillary controllers can enhance MPPI's efficiency and responsiveness, examining the impact on the overall solution. Furthermore, we demonstrated that integrating a parallelizable physics engine with MPPI's gradient-free approach makes it possible to generate real-time control inputs for robots in complex, contact-rich tasks with minimal modeling effort. We validated all results extensively through simulations and verified them with real-robot experiments. Below, we summarize the key findings of this thesis.

7.1.1 Interaction-Aware MPPI

In Chapter 3, we proposed a decentralized, communication-free, multi-agent MPPI algorithm with enhanced efficiency thanks to a novel two-stage sampling strategy and presented a discontinuous cost function that encourages compliance with navigation rules. First, thanks to MPPI's gradient-free nature, we performed collision avoidance via collision checking instead of Free Space Decomposition (FSD), vastly improving the success rate in navigation tasks in narrow maps compared to optimization-based MPC. Second, we demonstrated how interaction awareness, as in expecting cooperation and rule compliance from the other agents, is key to reducing collisions, rule violations, and travel time in urban canals. In this Chapter, however, we also discovered that the main obstacle to increasing the number of agents is the increased number of samples required to achieve an adequate approximation of the optimal control. As in previous work, we also found that MPPI, only sampling around a previous plan, is prone to failure in dynamic environments. Moreover, we have only used a constant velocity model to estimate the hidden goals of the other agents, which may not be a fair assumption in tight environments rich in interactions.

7.1.2 Learning-Based Trajectory Predictions

In Chapter 4, we used a centralized version of Interaction-Aware Model Predictive Path Integral (IA-MPPI) to generate a realistic artificial dataset of rule-abiding vessel interactions in urban canals. We then adapted and trained a trajectory prediction model based on Variational Recurrent Neural Network (VRNN) to estimate the future trajectories of the non-communicating agents. By extracting the agents' goals from predicted trajectories and integrating this information into the IA-MPPI controller, the planner's performance increased in environments with tight interactions compared to extracting such goals from a constant velocity model. Moreover, this approach outperformed treating the predicted trajectory as occupied space, leading to safer navigation in dense urban canals. However, limitations regarding the increasing number of samples required for systems that are larger or more difficult to control remain, as well as failures when the environment changes too quickly and unexpectedly.

7.1.3 Biased-MPPI

In Chapter 5, we investigated using multiple ancillary controllers to improve MPPI's sampling efficiency. We have mathematically shown how to arbitrarily modify the sampling distribution of MPPI and how doing so results in a bias in the approximation of the optimal control. We used classical and data-driven ancillary controllers, making the planner more reactive and efficient. Our experiments have shown that, with good ancillary controllers, we can more accurately swing up and track a reference with a rotational inverted pendulum and avoid collisions with obstacles thrown in the robot's path when driving at high speed. By adding ancillary controllers in complex multi-agent navigation scenarios, we achieved higher success rates while reducing the required samples tenfold.

7.1.4 Isaac-MPPI

In Chapter 6, we leveraged the gradient-free property of MPPI and used Isaac-Gym, a non-differentiable parallelizable simulator as a dynamic model. We significantly improved the sampling efficiency compared to standard MPPI not only by using ancillary controllers (e.g., sampling a zero velocity sequence) but also by implementing other tricks from the literature, such as using a Halton sequence instead of random Gaussian noise and fitting splines to the sampled sequences [113]. This allowed for real-time control of high-degree-of-freedom manipulators performing several contact-rich tasks with minimal modeling since we only relied on a Unified Robot Description Format (URDF) description of the robot and its environment. We performed non-prehensile manipulation tasks in simulation and with real robots with an omnidirectional base and a high-degree-of-freedom manipulator. We performed domain randomization in real time by randomizing the simulated environments, facilitating the real-world transfer. Our approach is highly generalizable to tasks such as whole-body pick and delivery with a mobile manipulator while significantly outperforming specialized non-prehensile manipulation baselines that require a substantial modeling effort.

7.2 Discussion

This thesis focused on interaction and rule-aware motion planning with MPPI for ASVs in urban canals. Although we made several advances to improve the behavior of ASVs in narrow, dynamic, and interaction-rich environments, such scenarios remain challenging. Moreover, while we improved upon MPPI and applied it to ground robots and manipulators, we also believe it will still take time before this controller can be widely adopted. In the following, we will discuss some key differences between MPPI and MPC, highlight the limitations of the approaches we propose, and discuss the remaining challenges.

7.2.1 MPPI or MPC

In the very first paper published during this project, which was not included in this thesis, we used an optimization-based MPC to drive an ASV in urban canals accounting for the navigation rules [8]. From then on, we solely focused on MPPI for motion planning. This begs the question: *What are the key benefits and drawbacks of MPPI over MPC?* We will attempt to answer this in the following paragraphs.

Planned trajectory and symmetries: Similarly to MPC, MPPI plans a sequence of inputs over a planning horizon. However, unlike MPC's input sequence that could theoretically be applied open-loop, MPPI's sequence should not. This stems from the fact that, under some assumptions, MPPI can be connected back to Stochastic Optimal Control (SOC) [10]. In the SOC framework, we have a system with Gaussian additive input noise. Let's imagine a symmetric problem where our robot has to drive either to the right or the left of an obstacle. Here, the optimum in the SOC setting is to drive straight toward the obstacle because the noise in the system will push the robot to either side. Replanning at each timestep, this symmetry is guaranteed to break [13].

Computational demand: Sampling input sequences, rolling-out trajectories, and computing their costs are all steps that can be parallelized in MPPI. We have demonstrated that even parallelizing on a CPU, MPPI can have five times longer planning horizons for systems with four times more inputs and states compared to an MPC solver generated with FORCES PRO [28]. We can take more samples on a GPU than on a CPU at the expense of a lower sequential speed and more CPU-GPU memory transfers. However, MPPI's performances plateaus quickly with increasing samples, thus requiring better sampling strategies [89]. Moreover, MPPI's massively parallel use of either CPU or GPU is more power-hungry than an optimization-based MPC, which may run on a single thread.

Optimality: An appropriate MPC solver will find the (local) optimum of the finite horizon optimal control problem if such a solution exists and the solver is run to convergence. Moreover, such a solution is guaranteed to satisfy all the constraints. While MPPI can be connected to SOC under some assumptions, it is only guaranteed to find its optimum when the number of samples approaches infinity. In practice, the quality of the solution is highly dependent on the sampling distribution. Moreover, for systems that are not input-affine, problems with arbitrary input costs, or arbitrary base and control measures, the connection to SOC is lost, and optimality can only be seen in the Information Theoretic sense [10]. Also, introducing exact state constraints in MPPI remains challenging [83].

Discontinuities: One of the main advantages of MPPI over MPC is that it is gradient-free. We leveraged this property to introduce discontinuous cost functions to embed navigation rules and perform collision avoidance by direct collision checking on the occupancy grid [28]. We also used this property when we employed a non-differentiable physics engine as a dynamic model and performed contact-rich tasks [85]. However, the cost function should still be fairly dense, and cost smoothness should still be preferred whenever possible as it provides a direction to exit the higher-cost region.

7.2.2 Future Work on Sampling-Based MPC

Although this dissertation presents significant advances in the theory and application of MPPI, there are still numerous opportunities to refine sampling-based MPC and explore new techniques. Below, we outline potential directions for further enhancing the performance of these methods and present new interesting applications.

MPPI and Reinforcement Learning: It was initially shown that policy improvements in Reinforcement Learning (RL) can be transformed into a path integral approximation problem, demonstrating performance improvement and better scalability compared to gradient-based policy learning [16]. RL can also be used to train a sampling policy for MPPI [83, 91]. While this can improve sampling efficiency in narrow tasks, it can hinder MPPI's generalizability. MPPI has also shown to be a good solver for model-based RL [21], and can be combined with model-free RL for value function approximation [120]. Recently, model-based RL, value function approximation, and sampling policy learning were combined with MPPI to train and deploy a single 317M parameter model to solve 80 different robotics tasks in different domains and with different embodiments [121, 122]. This type of work is extremely promising and provides an intriguing future research direction.

Sampling Efficiency and Model Uncertainty: In Chapter 5, we have demonstrated how one can change the sampling distribution employing several ancillary controllers to improve MPPI's reactivity and efficiency. While it can always be useful to include ancillary controllers to quickly converge to maneuvers such as breaking or keeping steady, many other ways exist to improve the overall sampling efficiency and exploration. Starting from a variational inference perspective, recent works perform sampling-based MPC using a Gaussian mixture [123], a particle representation [124] or a conditional normalizing flow [92] to approximate the posterior. These methods allow for more flexibility when representing the true distribution. Moreover, one can also modify the sampling to estimate model parameters online [80, 125]. Future work could continue improving sampling efficiency and exploration, which are key for performance and allow for online adaptation of model parameters.

Ego-conditioned prediction models: In Chapter 4, the prediction model we used outputs a joint trajectory for all agents in the scene, based only on the scene and past states of the agents. Another approach involves ego-conditioned prediction models, where the prediction is based on a proposed trajectory of the ego-agent. This allows the model to predict how other agents will react to specific actions taken by the ego-agent [126]. Incorporating such models into the planning loop could enhance interaction-awareness. Recent work, for example, integrates model predictive control (MPC) with a differentiable prediction model, enabling the ego-agent's plan to influence the predicted behavior of other agents [127]. However, such models' high computational cost of computing gradients prevents real-time performance. A promising direction for future research is developing fast, multi-query, ego-conditioned joint prediction models. Then, MPPI could generate several ego-agent plans, propagate them through the prediction model, and simultaneously obtain ego-conditioned predicted trajectories for all agents and plans. These trajectories could then be evaluated to compute an interaction-aware plan for the ego-agent.

7.2.3 Considerations and Vision on Unmanned Vessels

While numerous companies have begun testing unmanned autonomous cars on public roads, autonomous ships remain less visible. This disparity can be partly attributed to market size—car sales far exceed boat sales, and while cars are predominantly used for

daily commuting, private individuals commonly use boats for recreational purposes. This suggests that the market for autonomous cars is simply larger.

However, around 80% of global trade by volume is transported by ships, highlighting their crucial role in the global economy. If ships are such a significant component of worldwide commerce, why haven't they embraced full autonomy? Like passenger aircraft, large container ships already have partial autonomy, with autopilot systems handling much of the route under human supervision. Yet, they are definitely not unmanned.

Currently, large container ships operate with crews of 15 to 35 people. Approximately one-third of this crew is in the deck department, including the master, the officers, the bosun, and a few sailors. The remaining crew manages engine operations, electrical systems, maintenance, and provisions. Of those in the deck department, only a couple are actively involved in daily ship navigation at any given time, which already relies on autopilot systems. Thus, even if we achieve highly reliable autonomous navigation, including in challenging areas like ports or canal crossings such as Panama or Suez, the reduction in crew might be no more than 10%. This is far from realizing the goal of fully unmanned ocean-crossing ships.

The situation is different for smaller inland vessels. These often have crews of just one or two people, and autonomous navigation could, in theory, replace most of their tasks. However, as explored in this thesis, navigating crowded urban canals presents a unique challenge. Current motion planners still struggle in narrow, interaction-heavy environments. Moreover, there is a significant lack of high-quality datasets for marine environments, which hampers the development of accurate data-driven prediction models and perception systems.

7

Despite these challenges, autonomous vessels on inland waterways are within reach today. Projects like Roboat have shown that autonomy-capable hardware for vessels already exists. Furthermore, advancements in autonomous cars suggest that robust perception and obstacle classification technologies can be adapted for marine use. With the integration of state-of-the-art planning algorithms and strong engineering efforts, imagining a vessel navigating urban canals and adhering to navigation rules while operating with a high degree of caution is not too far-fetched.

Such a system may freeze during peak traffic in Amsterdam's busiest canals, but this would not prevent most of the vessel's intended tasks. For instance, high-density traffic is rarely an issue when ferrying passengers across large canals or fjords. In the case of garbage collection—one of the key use cases envisioned by the city of Amsterdam—autonomous vessels could transport floating garbage containers to collection centers during the night, entirely avoiding traffic.

With the added support of remote control, a few operators stationed centrally could manage an entire fleet of semi-autonomous vessels, performing various tasks such as deliveries or surveying in less congested areas or during times when interactions with traffic are minimal.

In conclusion, while fully autonomous vessels may not be feasible for all maritime operations today, particularly in dense urban environments, there are clear and immediate

opportunities for their deployment in controlled or low-traffic scenarios. With continued technological advancements and strategic integration of autonomy with human oversight, autonomous vessels can play a transformative role in inland waterways in the near future.

Bibliography

References

- [1] Stedelijke logistiek: de plannen voor de komende jaren. <https://amsterdam-autoluw-magazine.readz.com/editie-7-logistiek>. Accessed: 2024-09-20, in Dutch.
- [2] S Zhang, F Duarte, X Gui, L Johnsen, R Van De Ketterij, and C Ratti. Regaining amsterdam canals for waste collection. *MIT Senseable City Lab*, 2020.
- [3] Roboat - autonomy on the waterways. <https://roboat.tech/>. Accessed: 2024-09-20.
- [4] Zeabuz - navigate with confidence. <https://www.zeabuz.com/>. Accessed: 2024-09-20.
- [5] Wei Wang, David Fernández-Gutiérrez, Rens Doornbusch, Joshua Jordan, Tixiao Shan, Pietro Leoni, Niklas Hagemann, Jonathan Klein Schiphorst, Fabio Duarte, Carlo Ratti, and Daniela Rus. Roboat iii: An autonomous surface vessel for urban transportation. *Journal of Field Robotics*, 40(8):1996–2009, 2023.
- [6] Edmund F Brekke, Egil Eide, Bjørn-Olav H Eriksen, Erik F Wilthil, Morten Breivik, Even Skjellaug, Øystein K Helgesen, Anastasios M Lekkas, Andreas B Martinsen, Emil H Thyri, et al. milliamper: an autonomous ferry prototype. *Journal of Physics: Conference Series*, 2311(1):012029, 2022.
- [7] Yoshiaki Kuwata, Michael T. Wolf, Dimitri Zarzhitsky, and Terrance L. Huntsberger. Safe Maritime Autonomous Navigation With COLREGS, Using Velocity Obstacles. *IEEE Journal of Oceanic Engineering*, 39(1):110–119, January 2014. Conference Name: IEEE Journal of Oceanic Engineering.
- [8] Jitske de Vries, Elia Trevisan, Jules van der Toorn, Tuhin Das, Bruno Brito, and Javier Alonso-Mora. Regulations Aware Motion Planning for Autonomous Surface Vessels in Urban Canals. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3291–3297, May 2022.
- [9] Bjørn-Olav H. Eriksen, Morten Breivik, Erik F. Wilthil, Andreas L. Flåten, and Edmund F. Brekke. The branching-course model predictive control algorithm for maritime collision avoidance. *Journal of Field Robotics*, 36(7):1222–1249, 2019.
- [10] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, December 2018.

- [11] Yuezhe Zhang, Corrado Pezzato, Elia Trevisan, Chadi Salmi, Carlos Hernández Corbato, and Javier Alonso-Mora. Multi-modal mppi and active inference for reactive task and motion planning. *IEEE Robotics and Automation Letters*, 9(9):7461–7468, 2024.
- [12] Grady Robert Williams. *Model predictive path integral control: Theoretical foundations and applications to autonomous driving*. Doctoral Thesis, Georgia Institute of Technology, Atlanta, March 2019. Accepted: 2020-05-20T16:57:06Z Publisher: Georgia Institute of Technology.
- [13] H. J. Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11):P11011–P11011, November 2005. Publisher: IOP Publishing.
- [14] Hilbert J. Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. In *AIP Conference Proceedings*, volume 887, pages 149–181. AIP, February 2007.
- [15] Evangelos Theodorou, Yuval Tassa, and Emo Todorov. Stochastic differential dynamic programming. In *Proceedings of the 2010 American Control Conference, ACC 2010*, pages 1125–1132. IEEE Computer Society, 2010.
- [16] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.
- [17] Vicenç Gómez, Hilbert J Kappen, Jan Peters, and Gerhard Neumann. Policy search for path integral control. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I 14*, pages 482–497. Springer, 2014.
- [18] Thomas Mensink, Jakob Verbeek, and Bert Kappen. Ep for efficient stochastic control with obstacles. In *ECAI 2010*, pages 675–680. IOS Press, 2010.
- [19] Bart van den Broek, Wim Wiegerinck, and Bert Kappen. Stochastic optimal control of state constrained systems. *International Journal of Control*, 84(3):597–615, 2011.
- [20] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model Predictive Path Integral Control using Covariance Variable Importance Sampling. *CoRR*, September 2015.
- [21] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1714–1721, 2017.
- [22] Evangelos A. Theodorou and Emanuel Todorov. Relative entropy and free energy dualities: Connections to Path Integral and KL control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 1466–1473, December 2012. ISSN: 0743-1546.

- [23] Evangelos A Theodorou. Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations. *Entropy*, 17(5):3352–3375, 2015.
- [24] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1433–1440, 2016. Publisher: IEEE ISBN: 9781467380263.
- [25] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, May 2016.
- [26] Otton Nikodym. Sur une généralisation des intégrales de mj radon. *Fundamenta Mathematicae*, 15(1):131–179, 1930.
- [27] Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30(1):175–193, 1906.
- [28] Lucas Streichenberg, Elia Trevisan, Jen Jen Chung, Roland Siegwart, and Javier Alonso-Mora. Multi-Agent Path Integral Control for Interaction-Aware Motion Planning in Urban Canals. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1379–1385, May 2023.
- [29] Gemeente Amsterdam. Uitvoeringsplan Transport over Water, 2020. in Dutch.
- [30] Massachusetts Institute of Technology and Amsterdam Institute for Advance Metropolitan Solutions. Roboat project.
- [31] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties. Binnenvaartpolitiereglement, 2017. in Dutch.
- [32] Peter Trautman, Jeremy Ma, Richard M. Murray, and Andreas Krause. Robot navigation in dense human crowds: the case for cooperation. In *2013 IEEE International Conference on Robotics and Automation*, pages 2153–2160, May 2013. ISSN: 1050-4729.
- [33] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and Decision-Making for Autonomous Vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, May 2018.
- [34] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, March 1997.
- [35] 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, May 2008. ISSN: 1050-4729.

- [36] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-Body Collision Avoidance. In Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger, editors, *Robotics Research*, Springer Tracts in Advanced Robotics, pages 3–19, Berlin, Heidelberg, 2011. Springer.
- [37] Mikael Svenstrup, Thomas Bak, and Hans Jørgen Andersen. Trajectory planning for robots in dynamic human environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4293–4298, October 2010. ISSN: 2153-0866.
- [38] Jie Ji, Amir Khajepour, Wael William Melek, and Yanjun Huang. Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints. *IEEE Transactions on Vehicular Technology*, 66(2):952–964, February 2017.
- [39] 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, May 2017.
- [40] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P. How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350, September 2017. ISSN: 2153-0866.
- [41] 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, September 2020.
- [42] Mario Garzón and Anne Spalanzani. Game theoretic decision making for autonomous vehicles’ merge manoeuvre in high traffic scenarios. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3448–3453, October 2019.
- [43] Ran Tian, Nan Li, Ilya Kolmanovsky, Yildiray Yildiz, and Anouck R. Girard. Game-Theoretic Modeling of Traffic in Unsignalized Intersection Network for Autonomous Vehicle Control Verification and Validation. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2211–2226, March 2022.
- [44] Carlos E. Luis, Marijan Vukosavljev, and Angela P. Schoellig. Online Trajectory Generation With Distributed Model Predictive Control for Multi-Robot Motion Planning. *IEEE Robotics and Automation Letters*, 5(2):604–611, April 2020.
- [45] Mina Kamel, Javier Alonso-Mora, Roland Siegwart, and Juan Nieto. Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 236–243, September 2017. ISSN: 2153-0866.
- [46] Hai Zhu, Francisco Martinez Claramunt, Bruno Brito, and Javier Alonso-Mora. Learning Interaction-Aware Trajectory Predictions for Decentralized Multi-Robot

- Motion Planning in Dynamic Environments. *IEEE Robotics and Automation Letters*, 6(2):2256–2263, April 2021.
- [47] 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, October 2010. ISSN: 2153-0866.
- [48] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for Autonomous Cars that Leverage Effects on Human Actions. In *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016.
- [49] Weiwei Li and E. Todorov. Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system. *International Journal of Control - INT J CONTR*, 80:1439–1453, September 2007.
- [50] Evangelos Theodorou, Yuval Tassa, and Emo Todorov. Stochastic differential dynamic programming. In *Proceedings of the 2010 American Control Conference, ACC 2010*, pages 1125–1132. IEEE Computer Society, 2010.
- [51] Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model Predictive Path Integral Control: From Theory to Parallel Computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, February 2017. Publisher: American Institute of Aeronautics and Astronautics.
- [52] Wei Wang, Luis Mateos, Shinkyu Park, Pietro Leoni, Banti Gheneti, Fabio Duarte, Carlo Ratti, and Daniela Rus. Design, Modeling, and Nonlinear Model Predictive Tracking Control of a Novel Autonomous Surface Vehicle. In *IEEE International Conference on Robotics and Automation*. IEEE, May 2018.
- [53] Wei Wang, Banti Gheneti, Luis A. Mateos, Fabio Duarte, Carlo Ratti, and Daniela Rus. Roboat: An Autonomous Surface Vehicle for Urban Waterways. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6340–6347, November 2019. ISSN: 2153-0866.
- [54] Wei Wang, Tixiao Shan, Pietro Leoni, David Fernández-Gutiérrez, Drew Meyers, Carlo Ratti, and Daniela Rus. Roboat II: A Novel Autonomous Surface Vessel for Urban Environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1740–1747, October 2020. ISSN: 2153-0866.
- [55] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The Office Marathon: Robust navigation in an indoor office environment. In *2010 IEEE International Conference on Robotics and Automation*, pages 300–307, May 2010. ISSN: 1050-4729.
- [56] Tor Arne Johansen, Tristan Perez, and Andrea Cristofaro. Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment. *IEEE Transactions on Intelligent Transportation Systems*, 17(12):3407–3422, December 2016.

- [57] Walter Jansma, Elia Trevisan, Álvaro Serra-Gómez, and Javier Alonso-Mora. Interaction-Aware Sampling-Based MPC with Learned Local Goal Predictions, September 2023. Number: arXiv:2309.14931 arXiv:2309.14931 [cs].
- [58] Bruno Ferreira de Brito, Hai Zhu, Wei Pan, and Javier Alonso-Mora. Social-VRNN: One-Shot Multi-modal Trajectory Prediction for Interacting Pedestrians. In *Proceedings of the 2020 Conference on Robot Learning*, pages 862–872. PMLR, October 2021.
- [59] M. G. Mohanan and Ambuja Salgoankar. A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, 100:171–185, February 2018.
- [60] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart. Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots. In Alcherio Martinoli, Francesco Mondada, Nikolaus Correll, Grégory Mermoud, Magnus Egerstedt, M. Ani Hsieh, Lynne E. Parker, and Kasper Støy, editors, *Distributed Autonomous Robotic Systems*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 203–216. Springer, Berlin, Heidelberg, 2013.
- [61] Mario Zanon, Janick V. Frasch, Milan Vukov, Sebastian Sager, and Moritz Diehl. Model Predictive Control of Autonomous Vehicles. In Harald Waschl, Ilya Kolmanovsky, Maarten Steinbuch, and Luigi del Re, editors, *Optimization and Optimal Control in Automotive Systems*, Lecture Notes in Control and Information Sciences, pages 41–57. Springer International Publishing, Cham, 2014.
- [62] 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, April 2016. Publisher: Institute of Electrical and Electronics Engineers Inc.
- [63] 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.
- [64] David Fridovich-Keil, Andrea Bajcsy, Jaime F Fisac, Sylvia L Herbert, Steven Wang, Anca D Dragan, and Claire J Tomlin. Confidence-aware motion prediction for real-time collision avoidance¹. *The International Journal of Robotics Research*, 39(2-3):250–265, March 2020. Publisher: SAGE Publications Ltd STM.
- [65] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3399–3406, May 2018. ISSN: 2577-087X.
- [66] Christoforos Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld, and Jean Oh. Core Challenges of Social Robot Navigation: A Survey. *ACM Transactions on Human-Robot Interaction*, 12(3):36:1–36:39, April 2023.

- [67] 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, March 2015. Publisher: SAGE Publications Ltd STM.
- [68] Kaiwen Liu, Nan Li, H. Eric Tseng, Ilya Kolmanovsky, and Anouck Girard. Interaction-Aware Trajectory Prediction and Planning for Autonomous Vehicles in Forced Merge Scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 24(1):474–488, January 2023. Conference Name: IEEE Transactions on Intelligent Transportation Systems.
- [69] 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*, volume 28. Curran Associates, Inc., 2015.
- [70] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.
- [71] Elia Trevisan and Javier Alonso-Mora. Biased-mppi: Informing sampling-based model predictive control by fusing ancillary controllers. *IEEE Robotics and Automation Letters*, 9(6):5871–5878, 2024.
- [72] David Pérez-Morales and Vincent Fremont. Information-Theoretic Sensor-Based Predictive Control for Autonomous Vehicle Navigation: A Proof of Concept. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 879–884, September 2021.
- [73] Grady Williams, Brian Goldfain, Paul Drews, James M. Rehg, and Evangelos A. Theodorou. Best Response Model Predictive Control for Agile Interactions Between Autonomous Ground Vehicles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2403–2410, Brisbane, QLD, May 2018. IEEE.
- [74] Ihab S. Mohamed, Guillaume Allibert, and Philippe Martinet. Model Predictive Path Integral Control Framework for Partially Observable Navigation: A Quadrotor Case Study. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 196–203, December 2020.
- [75] Jintasit Pravitra, Evangelos A. Theodorou, and Eric N. Johnson. Flying complex maneuvers with model predictive path integral control. In *AIAA Scitech 2021 Forum*, pages 1–12. American Institute of Aeronautics and Astronautics Inc, AIAA, 2021.
- [76] Jintasit Pravitra, Kasey A. Ackerman, Chengyu Cao, Naira Hovakimyan, and Evangelos A. Theodorou. L1-Adaptive MPPI Architecture for Robust and Agile Control of Multirotors. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7661–7666, October 2020.

- [77] Vicenç Gómez, Sep Thijssen, Andrew Symington, Stephen Hailes, and Hilbert J Kappen. Real-time stochastic optimal control for multi-agent quadrotor systems. In *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, volume 2016, pages 468–476, March 2016.
- [78] Neng Wan, Aditya Gahlawat, Naira Hovakimyan, Evangelos A. Theodorou, and Petros G. Voulgaris. Cooperative Path Integral Control for Stochastic Multi-Agent Systems. In *2021 American Control Conference (ACC)*, pages 1262–1267, May 2021.
- [79] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D. Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation. In *5th Annual Conference on Robot Learning*, June 2021.
- [80] Ian Abraham, Ankur Handa, Nathan Ratliff, Kendall Lowrey, Todd D. Murphey, and Dieter Fox. Model-Based Generalization Under Parameter Uncertainty Using Path Integral Control. *IEEE Robotics and Automation Letters*, 5(2):2864–2871, April 2020.
- [81] Ermano Arruda, Michael J. Mathew, Marek Kopicki, Michael Mistry, Morteza Azad, and Jeremy L. Wyatt. Uncertainty averse pushing with model predictive path integral control. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 497–502, November 2017.
- [82] Lin Cong, Michael Grner, Philipp Ruppel, Hongzhuo Liang, Norman Hendrich, and Jianwei Zhang. Self-Adapting Recurrent Models for Object Pushing from Learning in Simulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5304–5310, October 2020.
- [83] Jan Carius, René Ranftl, Farbod Farshidian, and Marco Hutter. Constrained stochastic optimal control with learned importance sampling: A path integral approach. *The International Journal of Robotics Research*, page 02783649211047890, October 2021. Publisher: SAGE Publications Ltd STM.
- [84] Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo, December 2022.
- [85] Corrado Pezzato, Chadi Salmi, Elia Trevisan, Max Spahn, Javier Alonso-Mora, and Carlos Hernández Corbato. Sampling-based model predictive control leveraging parallelizable physics simulations. *IEEE Robotics and Automation Letters*, 10(3):2750–2757, 2025.
- [86] Grady Williams, Brian Goldfain, Paul Drews, Kamil Saigol, James Rehg, and Evangelos Theodorou. Robust Sampling Based Model Predictive Control with Sparse Objective Information. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018.
- [87] Thomas Mensink, Jakob Verbeek, and Bert Kappen. EP for efficient stochastic control with obstacles. In *Frontiers in Artificial Intelligence and Applications*, volume 215, pages 675–680. IOS Press, August 2010.

- [88] Masashi Okada and Tadahiro Taniguchi. Acceleration of Gradient-Based Path Integral Method for Efficient Optimal and Inverse Optimal Control. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3013–3020, May 2018.
- [89] Dylan M. Asmar, Ransalu Senanayake, Shawn Manuel, and Mykel J. Kochenderfer. Model Predictive Optimized Path Integral Strategies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3182–3188, May 2023.
- [90] Ihab S. Mohamed, Kai Yin, and Lantao Liu. Autonomous Navigation of AGVs in Unknown Cluttered Environments: Log-MPPI Control Strategy. *IEEE Robotics and Automation Letters*, 7(4):10240–10247, October 2022.
- [91] Raphael Kusumoto, Luigi Palmieri, Markus Spies, Akos Csiszar, and Kai. O. Arras. Informed Information Theoretic Model Predictive Control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2047–2053, May 2019.
- [92] Thomas Power and Dmitry Berenson. Variational Inference MPC using Normalizing Flows and Out-of-Distribution Projection. In *Robotics: Science and Systems XVIII*, volume 18, June 2022.
- [93] Isin M. Balci, Efstathios Bakolas, Bogdan Vlahov, and Evangelos A. Theodorou. Constrained Covariance Steering Based Tube-MPPI. In *2022 American Control Conference (ACC)*, pages 4197–4202, June 2022.
- [94] Ji Yin, Zhiyuan Zhang, Evangelos Theodorou, and Panagiotis Tsiotras. Trajectory Distribution Control for Model Predictive Path Integral Control using Covariance Steering. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1478–1484, May 2022.
- [95] Oktay Arslan, Evangelos A. Theodorou, and Panagiotis Tsiotras. Information-theoretic stochastic optimal control via incremental sampling-based algorithms. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 1–8, December 2014.
- [96] Manan Gandhi, Bogdan Vlahov, Jason Gibson, Grady Williams, and Evangelos A. Theodorou. Robust Model Predictive Path Integral Control: Analysis and Performance Guarantees. *IEEE Robotics and Automation Letters*, 6(2):1423–1430, April 2021. arXiv: 2102.09027.
- [97] Quanser Inc. QUBE - Servo 2 - Quanser.
- [98] Inés Tejado, Daniel Torres, Emiliano Pérez, and Blas M. Vinagre. Physical Modeling based Simulators to Support Teaching in Automatic Control: the Rotatory Pendulum. In *11th IFAC Symposium on Advances in Control Education ACE 2016*, volume 49, pages 75–80, January 2016.
- [99] K. J. Åström and K. Furuta. Swinging up a pendulum by energy control. *Automatica*, 36(2):287–295, February 2000.

- [100] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, January 2020.
- [101] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: an engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5):1327–1349, November 2021.
- [102] Francois R Hogan and Alberto Rodriguez. Reactive planar non-prehensile manipulation with hybrid model predictive control. *The International Journal of Robotics Research*, 39(7):755–773, June 2020.
- [103] Joao Moura, Theodoros Stouraitis, and Sethu Vijayakumar. Non-prehensile Planar Manipulation via Trajectory Optimization with Complementarity Constraints. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 970–976, Philadelphia, PA, USA, May 2022. IEEE.
- [104] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. 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, Brisbane, May 2018. IEEE.
- [105] Teguh Santoso Lembono, Antonio Paolillo, Emmanuel Pignat, and Sylvain Calinon. Memory of Motion for Warm-Starting Trajectory Optimization. *IEEE Robotics and Automation Letters*, 5(2):2594–2601, April 2020.
- [106] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.
- [107] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. Intelligent robotics and autonomous agents. MIT Press, Cambridge, Mass, 2nd ed edition, 2011.
- [108] Anqi Li, Mustafa Mukadam, Magnus Egerstedt, and Byron Boots. Multi-objective Policy Generation for Multi-robot Systems Using Riemannian Motion Policies. In Tamim Asfour, Eiichi Yoshida, Jaeheung Park, Henrik Christensen, and Oussama Khatib, editors, *Robotics Research*, Springer Proceedings in Advanced Robotics, pages 258–274, Cham, 2022. Springer International Publishing.
- [109] Nathan D. Ratliff, Karl Van Wyk, Mandy Xie, Anqi Li, and Muhammad Asif Rana. Generalized Nonlinear and Finsler Geometry for Robotics. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10206–10212, Xi’an, China, May 2021. IEEE.

- [110] Max Spahn, Martijn Wisse, and Javier Alonso-Mora. Dynamic Optimization Fabrics for Motion Generation. *IEEE Transactions on Robotics*, pages 1–16, 2023.
- [111] Giovanni Buizza Avanzini, Andrea Maria Zanchettin, and Paolo Rocco. Constrained model predictive control for mobile robotic manipulators. *Robotica*, 36(1):19–38, January 2018.
- [112] 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(1):269–296, May 2020.
- [113] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D. Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation. In *Proceedings of the 5th Conference on Robot Learning*, pages 750–759. PMLR, January 2022.
- [114] Michael Danielczuk, Arsalan Mousavian, Clemens Eppner, and Dieter Fox. Object Rearrangement Using Learned Implicit Collision Functions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6010–6017, Xi’an, China, May 2021. IEEE.
- [115] Jan Carius, René Ranftl, Farbod Farshidian, and Marco Hutter. Constrained stochastic optimal control with learned importance sampling: A path integral approach. *The International Journal of Robotics Research*, 41(2):189–209, February 2022.
- [116] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Vilamoura-Algarve, Portugal, October 2012. IEEE.
- [117] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari. FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 93(1):13–29, January 2020.
- [118] Max Spahn, Chadi Salmi, and Javier Alonso-Mora. Local Planner Bench: Benchmarking for Local Motion Planning, 2022.
- [119] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, June 2011.
- [120] Mohak Bhardwaj, Sanjiban Choudhury, and Byron Boots. Blending {mpc} & value function approximation for efficient reinforcement learning. In *International Conference on Learning Representations*, 2021.
- [121] Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8387–8406. PMLR, 17–23 Jul 2022.

- [122] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024.
- [123] Masashi Okada and Tadahiro Taniguchi. Variational Inference MPC for Bayesian Model-based Reinforcement Learning. In *Proceedings of the Conference on Robot Learning*, pages 258–272. PMLR, May 2020.
- [124] Alexander Lambert, Fabio Ramos, Byron Boots, Dieter Fox, and Adam Fishman. Stein Variational Model Predictive Control. In *Proceedings of the 2020 Conference on Robot Learning*, pages 1278–1297. PMLR, October 2021.
- [125] Lucas Barcelos, Alexander Lambert, Rafael Oliveira, Paulo Borges, Byron Boots, and Fabio Ramos. Dual Online Stein Variational Inference for Control and Dynamics. In *Robotics: Science and Systems XVII*, volume 17, July 2021.
- [126] Ekaterina Tolstaya, Reza Mahjourian, Carlton Downey, Balakrishnan Vadarajan, Benjamin Sapp, and Dragomir Anguelov. Identifying driver interactions via conditional behavior prediction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3473–3479, 2021.
- [127] Piyush Gupta, David Isele, Donggun Lee, and Sangjae Bae. Interaction-aware trajectory planning for autonomous vehicles with analytic integration of neural networks into model predictive control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7794–7800, 2023.

Glossary

APF Artificial Potential Fields.

ASV Autonomous Surface Vessel.

COLREGs COLLision avoidance REGulations at sea.

CVM Constant Velocity Model.

DDP Differential Dynamic Programming.

DOF Degrees of Freedom.

DWA Dynamic Window Approach.

FSD Free Space Decomposition.

HJB Hamilton-Jacobi-Bellman.

IA-MPPI Interaction-Aware Model Predictive Path Integral.

iLQG iterative Linear Quadratic Gaussian.

KL Kullback-Leibler.

LBM Learning-Based Model.

MPC Model Predictive Control.

MPPI Model Predictive Path Integral control.

ORCA Optimal Reciprocal Collision Avoidance.

PDE Partial Differential Equation.

PRM Probabilistic Roadmap.

RL Reinforcement Learning.

ROS Robot Operating System.

RRT Rapidly exploring Random Tree.

RVO Reciprocal Velocity Obstacles.

SOC Stochastic Optimal Control.

URDF Unified Robot Description Format.

VO Velocity Obstacles.

VRNN Variational Recurrent Neural Network.

Curriculum Vitæ

Elia Trevisan

26/11/1995 Born in Pordenone, Italy.

Education

2020-2024 PhD in Robotics
Delft University of Technology, Netherlands.
Thesis: Model Predictive Path Integral Control for Interaction-Rich Local Motion Planning in Dynamic Environments.

2018-2020 MSc in Systems and Control, *Cum Laude*
Delft University of Technology, Netherlands.
Thesis: Joint Estimation of Object and Aberration for High Numerical Aperture Microscopy.

2014-2018 BSc in Automation Engineering
University of Bologna, Italy, and Tongji University, China.
Thesis: Automated Testing of Household Appliances with a Robot Manipulator in ROS.
Award: AlmaTong double-degree scholarship.

2009-2014 Electrical Engineering High School Diploma
ITST J.F. Kennedy, Italy.

Experience

2019-2020 Teaching Assistant
Delft University of Technology, Netherlands
For three master courses: Introduction Project, Nonlinear System Theory, and Integration Project Systems and Control.

2019-2020 Board Member and Event Manager
Delft Students Association Kalman, Netherlands

2018-2018 PLC Software Engineer
Tecnoquadri, Italy

List of Publications

Journals

- 1. C. Pezzato*, C. Salmi*, **E. Trevisan***, M. Spahn, J. Alonso-Mora and C. H. Corbato, "Sampling-based Model Predictive Control Leveraging Parallelizable Physics Simulations," IEEE Robotics and Automation Letters (RA-L), January 2025.
- 2. Y. Zhang, C. Pezzato, **E. Trevisan**, C. Salmi, C. H. Corbato and J. Alonso-Mora, "Multi-Modal MPPI and Active Inference for Reactive Task and Motion Planning," IEEE Robotics and Automation Letters (RA-L), June 2024.
- 3. **E. Trevisan** and J. Alonso-Mora, "Biased-MPPI: Informing Model Predictive Path Integral Control by Fusing Ancillary Controllers," IEEE Robotics and Automation Letters (RA-L), May 2024.

Conference Proceedings

- 1. W. Jansma, **E. Trevisan**, A. Serra-Gómez and J. Alonso-Mora, "Interaction-Aware Sampling-Based MPC with Learned Local Goal Predictions," 2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), Boston, MA, USA, 2023. **Best paper award finalist.**
- 2. L. Streichenberg*, **E. Trevisan***, J. J. Chung, R. Siegwart and J. Alonso-Mora, "Multi-Agent Path Integral Control for Interaction-Aware Motion Planning in Urban Canals," 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023.
- 3. 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," 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 2022.

Workshop Papers

- 1. C. Pezzato*, C. Salmi*, **E. Trevisan***, J. Alonso-Mora and C. H. Corbato, "Sampling-Based MPC Using a GPU-parallelizable Physics Simulator as Dynamic Model: an Open Source Implementation with IsaacGym," Embracing Contacts - Workshop at ICRA 2023. **Best paper award finalist.**

In Preparation

- 1. **E. Trevisan***, K. Mustafa*, G. Notten, X. Wang, and J. Alonso-Mora, "Dynamic Risk-Aware MPPI for Mobile Robots in Crowds via Efficient Monte Carlo Approximations".
- 2. J. van der Saag, **E. Trevisan**, W. Falkena, J. Alonso-Mora, "Active Disturbance Rejection Control (ADRC) for Trajectory Tracking of a Seagoing USV: Design, Simulation, and Field Experiments".
- 3. **E. Trevisan**, A. Gonzalez-Garcia, J. Swevers, and J. Alonso-Mora, "Interaction-Aware Model Predictive Path Integral Control for Motion Planning in Urban Settings".

* Indicates equal contribution.

1. Included in this thesis.

