

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Luka Petrović

HIGH-DIMENSIONAL TRAJECTORY OPTIMIZATION FOR ROBOT MOTION PLANNING BASED ON GAUSSIAN PROCESSES

DOCTORAL THESIS

Zagreb, 2022



University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Luka Petrović

HIGH-DIMENSIONAL TRAJECTORY OPTIMIZATION FOR ROBOT MOTION PLANNING BASED ON GAUSSIAN PROCESSES

DOCTORAL THESIS

Supervisor: Associate Professor Ivan Marković, PhD

Zagreb, 2022



Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Luka Petrović

OPTIMIZACIJA VISOKODIMENZIJSKIH TRAJEKTORIJA ZA PLANIRANJE GIBANJA ROBOTA ZASNOVANA NA GAUSSOVIM PROCESIMA

DOKTORSKI RAD

Mentor: izv. prof. dr. sc. Ivan Marković

Zagreb, 2022.

Doctoral thesis was written at the University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering.

Supervisor: Associate Professor Ivan Marković, PhD

Thesis contains 135 pages

Thesis no.:

ABOUT THE SUPERVISOR

IVAN MARKOVIĆ was born in Osijek, Croatia in 1985. He finished a math-oriented high school in Osijek in 2003 and received the master and PhD degrees in Eectrical Engineering under the mentorship of Prof. Ivan Petrović from the University of Zagreb Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia in 2008 and 2014, respectively.

In 2008 he was employed by S.C.A.N. Ltd. where he worked on developing control algorithms and supervision interfaces for terminal automation systems. In 2009 he joined FER and the Department of Control and Computer Engineering as a research assistant where he enrolled the PhD studies in the same year. In 2009 and 2012 he was a member of the National Organizing Committee and in 2022 the Programe Committee Chair of international scientific conferences. In 2009 he became the coordinator of the technical editing team of the journal "Automatika – Journal for Control, Measurement, Electronics, Computing and Communications" and since 2018 he serves as the deputy Editor-In-Chief. In collaboration with Siemens Croatia Inc. he has been a course trainer for programmes developed for industry participants and in 2018. he recieved the recognition for his contribution in advancing education for SIMATIC automation systems. He actively participated as a researcher or work package leader in 6 national and 7 international research projects. During 2013 and 2014 he was a visiting researcher in the laboratory led by Prof. Francois Chaumette at INRIA Rennes-Bretagne Atlantique u Rennesu, France. He published 23 journal papers and more than 40 conference papers in the fields of estimation theory, sensor fusion, and robot vision with applications to autonomous mobile robot and vehicle control.

He is a member of the Institute of Electrical and Electronics Engineers (IEEE), Technical Committee for Robotics of the International Federation of Automatic Control (IFAC), and the Croatian Society for Communication, Computing, Electronics, Measurement and Control (KoREMA). In 2018 he received the young scientist award "Vera Johanides" of the Croatian Academy of Engineering, in 2014 silver plaque "Josip Lončar" of FER for an outstanding doctoral dissertation, in 2008 the "INETEC award" of the Institute of Nuclear Technology for academic excellence during the undergraduate and graduate studies, and in 2006 the "Josip Lončar" award of FER for student excellence in the respective academic year.

O MENTORU

IVAN MARKOVIĆ rođen je 1985. godine u Osijeku. Treću gimnaziju u Osijeku završio je 2003. godine te je diplomirao i doktorirao u polju elektrotehnike pod mentorstvom prof. dr. sc. Ivana Petrovića na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER), 2008. odnosno 2014. godine.

Po završetku studija zapošljava se u tvrtki S.C.A.N. d.o.o. gdje radi na razvijanju algoritama upravljanja i nadzora u sustavima automatizacije naftnih terminala. U 2009. godini zapošljava se u Zavodu za automatiku i računalno inženjerstvo FER-a u zvanju znanstvenog novaka u suradničkom zvanju asistenta te iste godine upisuje i doktorske studije. U 2009. i 2012. godini bio je član organizacijskog odbora, a u 2022. predsjednik programskog odbora međunarodnih znanstvenih konferencija. Od 2009. godine koordinator je tehničkog uredništva časopisa "Automatika – časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije", a od 2018. je zamjenik glavnog i odgovornog urednika. U suradnji s tvrtkom Siemens Hrvatska d.d. bio predavač je na tečajevima za polaznike iz industrije te je 2018. godine primio priznanje za poseban doprinos unaprijeđenu nastave i edukaciji korištenjem SIMATIC automatizacijskih sustava. Aktivno je sudjelovao kao suradnik ili voditelj radnog paketa na 6 nacionalnih i 7 međunarodnih znanstvenih projekata. Tijekom 2013. i 2014. godine bio je gostujući istraživač u grupi prof. dr. sc. Francoisa Chaumettea na ustanovi INRIA Rennes-Bretagne Atlantique u Rennesu, Francuska. Objavio je 23 znanstvena rada u časopisima i više od 40 znanstvenih radova u zbornicima skupova u području estimacije, fuzije senzora i robotskog vida s primjenom u upravljanju autonomnim mobilnim robotima i vozilima.

Član je društva Institute of Electrical and Electronics Engineers (IEEE), tehničkog odbora za robotiku društva International Federation of Automatic Control (IFAC) i Hrvatskog društva za komunikacije, računarstvo, elektroniku, mjerenja i automatiku (KoREMA). Godine 2018. primio je nagradu mladom znanstveniku "Vera Johanides" Akademije tehničkih znanosti Hrvatske, 2014. srebrnu plaketu "Josip Lončar" FER-a za posebno istaknutu doktorsku disertaciju, 2008. "INETEC nagradu" Instituta za nuklearnu tehnologiju za izvrsnost postignutu tijekom studija, a 2006. godine nagradu "Josip Lončar" FER-a za izvanredan uspjeh u toj akademskoj godini.

ABSTRACT

Motion planning is one of the fundamental tasks in robotics with the goal of producing feasible robot trajectories in configuration space that reach a desired goal optimally according to a given criterion. The rapid growth in complexity of both robots and their operating environments has accentuated the need for reliable high-dimensional motion planning. The principal purpose of this thesis is the development of novel high-dimensional motion planning methods that enable safe and efficient robot operation in real-world environments. We rely on trajectory optimization in order to produce robot trajectories that are smooth, collision-free and satisfy arbitrary task-dependent criteria. A common theme throughout this thesis is reliance on continuous-time Gaussian processes as trajectory representations that underpin efficient collision-checking and generate smooth robot trajectories.

The trajectory optimization approach to motion planning results in computationally efficient methods for robots with many degrees of freedom, but these methods are often unable to find a feasible solution in complex environments due to getting stuck in local minima. One of the goals of this thesis is to develop methods that ameliorate the local minima problem while retaining other desirable properties of trajectory optimization, such as computational efficiency and trajectory smoothness. With this in mind, this thesis first introduces heteroscedastic Gaussian process priors crafted for improving collision avoidance. Coupled with the proposed cross-entropy based stochastic trajectory optimization, the resultant motion planning method has the ability to plan collision-free trajectories even for high-dimensional robots in cluttered environments. Besides collision avoidance, there often exist multiple different costs and constraints that have to be taken into account during planning, for example adhering to joint and velocity limits, kinematic singularity avoidance and torque minimization. To this end, this thesis proposes mixtures of Gaussian processes as trajectory representations that can generate multiple solution trajectories and a stochastic gradient estimation method supporting the inclusion of non-differentiable optimization costs. Finally, deploying robots in real-world environments with human presence requires additional considerations regarding perception, mapping and human trajectory prediction due to limited onboard resources. This thesis presents an integrated motion planning, perception and human trajectory prediction framework that utilizes a single robot-mounted RGB-D camera to achieve predictive real-time motion planning in dynamic environments.

KEYWORDS: high-dimensional motion planning, trajectory optimization, Gaussian processes, collision avoidance, singularity avoidance, stochastic optimization, cross-entropy optimization, mixtures of Gaussian processes, robotic arm, mobile manipulator

SAŽETAK

Optimizacija visokodimenzijskih trajektorija za planiranje gibanja robota zasnovana na Gaussovim procesima

Planiranje gibanja predstavlja jedan od temeljnih problema u robotici. Uspješno planiranje gibanja neophodno je za postizanje učinkovitog djelovanja robota te njegove sigurne interakcije u stvarnim okruženjima, neovisno radi li se o mobilnome robotu koji navigira u ravnini, robotskoj ruci koja manipulira objektima ili o složenijem robotskom sustavu koji istovremeno pokušava oboje. Metode planiranja gibanja pokušavaju generirati trajektorije u konfiguracijskom prostoru robota koje su ostvarive i optimalno dostižu željeni cilj prema zadanome kriteriju koji ovisi o zadatku, robotu ili okruženju u kojemu robot djeluje. Ostvarivost je u pravilu podudarna zadovoljavanju ograničenja poput dolaska do željenog cilja bez sudara s okolinom, dok se optimalnost obično odnosi na glatkoću trajektorije, minimiziranje akceleracije, duljine putanje ili neke druge kvalitativne metrike. Računska učinkovitost također je vrlo važna, budući da je ponavljanje planiranja trajektorije potrebno u dinamičkim okruženjima sa stalnim promjenama.

Posljednjih se godina dogodio značajan napredak u istraživanju i primjenu planiranja gibanja za robote s manje od sedam stupnjeva slobode, rezultirajući mobilnim robotima koji prenose teret u skladištu, industrijskim robotskim rukama koje obavljaju zadatke poput rezanja i varenja te bespilotnim letjelicama u zadacima nadzora. Iako roboti postaju sveprisutni kada se radi o takvim jasno definiranim zadacima koji ne zahtijevaju interakciju s ljudima, postavlja se pitanje zašto trenutno ne postoje roboti koji bi nam pomogli u svakodnevnim zadacima u našim domovima ili uredima. Gdje je robot koji će sastaviti komad namještaja ili oprati posuđe? Kratak je odgovor da za složene zadatke trebamo složene robotske sustave. Čak i za zadatke koji su ljudima jednostavni trebamo složene robotske sustave. Ljudsko tijelo ima više od dvjesto stupnjeva slobode gibanja. Ako težimo robotima koji mogu replicirati određene aspekte ljudskog ponašanja, trebamo složenije robote te posljedično i metode koje će uspješno planirati gibanje takvih robota. Robot kojemu je cilj donijeti šalicu kave osobi u drugoj prostoriji treba imati mobilnu bazu za navigaciju između prostorija, ruku za za držanje šalice te potencijalno dodatnu ruku za otvaranje vrata. Čak i tako jednostavan zadatak zahtijeva planiranje gibanja za sustav s više od deset stupnjeva slobode, što je često neizvedivo u stvarnome vremenu s postojećim metodama. Očito je da postoji nesrazmjer između sposobnosti robota koje zamišljamo i planova gibanja koje u stvarnome vremenu možemo dobiti metodama koje predstavljaju stanje tehnike. Iz tog se razloga u disertaciji bavimo problemom planiranja gibanja u visokodimenzijskim prostorima.

Tri najčešća načina za rješavanje problema planiranja gibanja u visokodimenzijskim prostorima su redom metode zasnovanje na mrežastoj karti prostora, metode zasnovane na uzorkovanju te metode optimizacije trajektorija. Budući da su metode zasnovane na prikazu prostora mrežastom kartom kompletne, one se često koriste za planiranje gibanja u okruženjima nalik labirintu te u okruženjima s uskim prolazima. Ključan je nedostatak tih metoda prokletstvo dimenzionalnosti zbog čega postaju previše računski zahtjevni za visokodimenzijske sustave poput humanoidnih robota i mobilnih manipulatora. Metode zasnovane na uzorkovanju vjerojatnosno su kompletne što znači da se vjerojatnost da će dati rješenje približava iznosu jedan što se duže metoda izvodi. Njihov je nedostatak činjenica da dobivene putanje često uključuju redundantno gibanje te stoga zahtijevaju naknadu obradu kako bi se generirane trajektorije izgladile i skratile. Iako metode zasnovane na mrežastoj karti prostora i metode zasnovane na uzorkovanju mogu *uvijek* pronaći rješenje visokodimenzijskih problema, to i dalje ne mogu pouzdano činiti u stvarnome vremenu, posebice za probleme sa složenim ograničenjima ili u dinamičkim okruženjima. Iz tog su razloga predložene metode optimizacije trajektorija čija se složenost ne skalira eksponencijalno s brojem stupnjeva slobode robota. Međutim, metode optimizacije trajektorija pronalaze lokalno optimalno rješenje, što implicira da u pojedinim slučajevima neće uspjeti pronaći izvedivo rješenje. U ovoj se disertaciji oslanjamo na optimizaciju trajektorija kao pristup planiranju gibanja u visokodimenzijskim prostorima, smatrajući da ublažavanje problema lokalnih minimuma dovoljno za pouzdan rad u stvarnim okruženjima predstavlja jednostavniji izazov od rješavanja inherentnih nedostataka metoda zasnovanih na mrežastoj karti prostora i metoda zasnovanih na uzorkovanju. Metode optimizacije trajektorija dobro se skaliraju na visokodimenzijske prostore, generiraju glatke trajektorije te mogu uzeti u obzir različite kriterijske funkcije i ograničenja. Zahvaljujući računskoj efikasnosti prikladne su za planiranje gibanja u dinamičkim okruženjima.

Metode optimizacije trajektorija najčešće se oslanjaju na diskretne trajektorije robota, odnosno, generiraju trajektorije koje su predstavljene skupovima stanja robota u diskretnim vremenskim trenucima. Međutim, ako je trajektorija predstavljena malim brojem diskretnih stanja, moguće je previdjeti tanke prepreke u prostoru koje se nalaze između dva diskretna stanja robota što može uzrokovati sudar. Također, trajektorije s malim brojem diskretnih stanja često zahtijevaju dodatnu naknadnu obradu kako bi ih robot mogao izvesti. S druge strane, ako je trajektorija gusto diskretizirana, optimizacija može postati nepotrebno računski zahtjevna. Iz navedenih se razloga u svim znanstvenim doprinosima oslanjamo na predstavljanje trajektorija robota kontinuiranim GP-ima, koji čine matematički utemeljen način za predstavljanje kontinuiranih trajektorija malim brojem diskretnih stanja s mogućnošću efikasne interpolacije stanja u bilo kojem vremenskom trenutku, čime zaobilaze nedostatke diskretnih trajektorija. U nastavku su ukratko opisani glavni znanstveni doprinosi disertacije.

#1 Metoda za stohastičku optimizaciju visokodimenzijske trajektorije robota zasnovana na unakrsnoj entropiji i heteroskedastičnim Gaussovim procesima.

Prvi se doprinos odnosi na novu metodu planiranja trajektorije zasnovane na stohastičkoj optimizaciji s ciljem pronalaska trajektorije koja nije u sudaru s okolinom, a generirana je iz kontinuiranog Gaussovog procesa (GP). Uvođenje heteroskedastičnosti

u navedeni GP u kombinaciji s estimacijom matrice kovarijance koja ima rijetku strukturu omogućuje generiranje *a priori* GP-a koji je prikladniji za izbjegavanje sudara. Predložena metoda stohastičke optimizacije zasnovana na unakrsnoj entropiji omogućuje optimizaciju trajektorija zasnovanu na uzorkovanju važnosti koja omogućuje bolje pretraživanje prostora rješenja u kompleksnim okruženjima u odnosu na metode koje predstavljaju trenutno stanje tehnike.

#2 Metoda optimizacije visokodimenzijskih trajektorija zasnovana na stohastičkoj estimaciji gradijenta korištenjem mješavine Gaussovih procesa koja omogućuje uključivanje više složenih kriterija.

Drugi se doprinos sastoji od novoga načina predstavljanja trajektorija kao uzoraka iz mješavine kontinuiranih GP-a te pridružene metode za stohastičku estimaciju gradijenta koja osvježava parametre mješavine na način koji minimizira kriterijsku funkciju. Prednosti predložene metode očituju se u sposobnosti da uspješno savlada probleme planiranja gibanja u visokodimenzijskim prostorima uz očuvanje glatkoće trajektorije, mogućnosti da pronađe više od jedne trajektorije koja nije u sudaru s okolinom te mogućnosti za uključivanje kriterijskih funckija koje nisu diferencijabilne.

#3 Inkrementalna metoda zaključivanja na faktor grafu za istovremeno praćenje gibajućih objekata i planiranje gibanja robota zasnovana na Gaussovim procesima.

Treći doprinos obuhvaća algoritam istovremenog planiranja gibanja i praćenja gibajućih prepreka, algoritam za predikciju trajektorije čovjeka koja u obzir uzima kontekst te praktičan okvir za percepciju i planiranje gibanja koji omogućuje integraciju navedenih algoritama u jedinstvenu cjelinu. Navedeni okvir omogućuje prediktivno planiranje gibanja cijeloga tijela mobilnog manipulatora u stvarnim dinamičkim okruženjima koje uzima u obzir predviđenje trajektorije gibajućih prepreka, uključujući ljudi.

Disertacija je podijeljena u sedam poglavlja. Svako poglavlje započinje kratkim sažetkom s ciljem koncizne prezentacije metode opisane u poglavlju i doprinosa koje ona donosi. Nakon sažetka čitatelja postupno upućujemo u rješavani problem te elaboriramo motivaciju za predloženo rješenje. Na kraju svakoga poglavlja nalazi se kratak pregled u kojem se još jednom spominju ključni doprinosi poglavlja, postignuti eksprimentalni rezultati te zaključci koji iz njih proizlaze. U prvome poglavlju dajemo uvod u disertaciju, predstavljajući motivaciju i ističući probleme koje rješavamo. Također opisujemo predložene znanstvene doprinose i oblik disertacije, zajedno s kratkim sažetkom svakog poglavlja.

U drugome poglavlju dajemo pregled postojećih metoda za planiranje gibanja robota u visokodimenzijskim prostorima. Poglavlje počinjemo prezentacijom formulacije problema te predstavljamo neke od prvih pokušaja rješavanja iz literature. Nakon toga, poglavlje pruža strukturirani uvid u tri najraširenija načina za rješavanje problema planiranja gibanja, a to su redom pristupi zasnovani na mrežastoj karti prostora, pristupi zasnovani na uzorkovanju te pristupi zasnovani na optimizaciji trajektorija. Budući da svaka metoda koju predlažemo u sklopu ove disertacije pripada pristupima optimizacije trajektorija, iste dodatno dijelimo

na pristupe zasnovane na gradijentu i pristupe bez gradijenta te podrobnije naglašavamo i opisujemo relevantne metode iz navedenih područja.

U trećem poglavlju predstavljamo temeljnu teoretsku podlogu i opisujemo zajedničke koncepte koji su relevantni kroz cijelu disertaciju. Prvo opisujemo predstavljanje robotskih trajektorija korištenjem kontinuiranih GP-a prikladnih za planiranje gibanja robota, što čini temelj svake od predloženih metoda. Nadalje, opisujemo korišteni model robota i način predstavljanja okruženja koje koristimo za provjeru sudara robota i okoline u svim eksperimentima u disertaciji. Na kraju poglavlja predstavljamo problem kinematičkih singulariteta, zajedno s novim indeksom singulariteta koji poštuje geometriju prostora. Navedeni indeks kasnije koristimo u sklopu jedne od predloženih metoda planiranja gibanja za identifikaciju i izbjegavanje singulariteta korištenjem optimizacije trajektorija.

Četvrto se poglavlje dotiče nove metode za visokodimenzijsko planiranje gibanja robota u kompleksnim okruženjima. Uvodimo heteroskedastične kontinuirane GP-e te predstavljamo njihove prednosti u smislu veće učinkovitosti izbjegavanja sudara u kompleksnim okruženjima. Slijedi izvod stohastičke metode optimizacije trajektorija zasnovane na unakrsnoj entropiji koja omogućuje bolje pretraživanje prostora rješenja u odnosu na trenutno stanje tehnike uz konkurentno vrijeme izvođenja. Predložena je metoda zatim evaluirana u tri simulacijska scenarija koji za cilj imaju demonstraciju poboljšanja učinkovitosti izbjegavanja sudara te mogućnost razvijene metode da riješi problem planiranja trajektorije robota u visokodimenzijskim prostorima.

U petome poglavlju prezentiramo metodu za višekriterijsko planiranje gibanja zasnovano na mješavinama GP-a te stohastičkoj estimaciji gradijenta. U početku opisujemo predloženo predstavljanje trajektorija mješavinama te prednosti koje iz njih slijede, poput mogućnosti generiranja više ostvarivih rješenja i omogućavanja boljeg pretraživanja prostora trajektorija u odnosu na predstavljanje trajektorija jednim GP-om. Zatim predstavljamo algoritam za stohastičku estimaciju gradijenta koji omogućuje uključivanje nediferencijabilnih optimizacijskih kriterija. Potom provodimo detaljnu evaluaciju metode koja uključuje višestruke usporedbe u simulacijama s različitim robotima te eksperimente sa stvarnim robotima u laboratoriju. Na kraju raspravljamo o dobivenim rezultatima, objašnjavajući visoke performanse metode u izbjegavanju sudara, minimizaciji momenata motora, izjegavanju kinematičkih singulariteta uz zadržavanje glatkoće trajektorije i računske učinkovitosti.

U šestome je poglavlju prezentiran novi integrirani okvir za percepciju i planiranje gibanja koji uzima u obzir predviđeno gibanje čovjeka, omogućujući prediktivno planiranje gibanja u stvarnim dinamičkim okruženjima. Prvo dajemo pregled relevantnih metoda za percepciju robota u dinamičkim okruženjima te predikciju trajektorije čovjeka, nakon čega predstavljamo predloženi okvir. Zatim opisujemo novi skup podataka za predviđanje gibanja ljudi te pojedinačno predstavljamo svaku komponentu navedenog integriranog okvira, opisujući trodimenzionalno mapiranje u stvarnome vremenu, novu metodu za predikciju trajektorije čovjeka te metodu planiranja gibanja robota zasnovanu na optimizaciji trajektorija koja podržava rad u stvarnome vremenu. Rezultantni okvir validiran je u višestrukim eksperimentima s mobilnim manipulatorom s jednim RGB-D senzorom u stvarnome okruženju čime je demonstrirana sposobnost za prediktivno planiranje gibanja robota s visokim stupnjem slobode u okruženjima s ljudskom prisutnošću. Na kraju poglavlja prezentirana je alternativna formulacija optimizacije trajektorija zasnovana na

istovremenom planiranju gibanja i predikciji trajektorije čovjeka.

Posljednje, sedmo poglavlje disertacije daje zaključke o predloženim znanstvenim doprinosima te predlaže nekoliko potencijalnih smjerova i ideja za budući rad u području optimizacije trajektorija za planiranje gibanja robota u visokodimenzijskim prostorima.

KLJUČNE RIJEČI: visokodimenzijsko planiranje gibanja, optimizacija trajektorija, Gaussovi procesi, izbjegavanje sudara, izbjegavanje singulariteta, stohastička optimizacija, optimizacija unakrsne entropije, mješavine Gaussovih procesa, robotska roka, mobilni manipulator

CONTENTS

1 INTRODUCTION 1
1.1 Motivation and problem statement 1
1.2 Original contributions 3
1.3 Outline of the thesis 4

2 AN OVERVIEW OF MOTION PLANNING METHODS 6
2.1 Introduction 6
2.2 Grid-based approaches 8
2.3 Sampling-based methods 9
2.4 Trajectory optimization 11
2.4.1 Gradient-based methods 13
2.4.2 Stochastic optimization methods 14
2.5 Summary 16

3 THEORETICAL BACKGROUND 17
3.1 Gaussian processes as trajectory representations 17
3.1.1 GP prior for motion planning 18
3.1.2 GP interpolation 19
3.1.3 Constant-velocity motion model 20
3.2 Collision avoidance 21
3.2.1 Robot model 21
3.2.2 Environment representation 22
3.2.3 Collision checking 23
3.3 Kinematic singularity avoidance 24
3.3.1 Kinematic singularities 25
3.3.2 Manipulability index 26
3.3.3 Dexterity index 27
3.3.4 A geometry-aware singularity index 27
3.4 Summary 29

4 MOTION PLANNING IN CLUTTERED ENVIRONMENTS 30
4.1 Introduction 30
4.2 Heteroscedastic GPs as trajectory representations 31
4.2.1 Heteroscedastic GP prior for motion planning 32
4.2.2 Benefits of proposed heteroscedasticity 34
4.3 Cross-entropy based trajectory optimization 34

4.3.1 Problem formulation . . . 35
4.3.2 Cross-entropy optimization method . . . 35
4.3.3 Updating the mean and the covariance . . . 37
4.3.4 Computational Efficiency Remarks . . . 39
4.4 Results . . . 41
4.4.1 The Maze Benchmark . . . 42
4.4.2 The Robot Arm Planning Benchmark . . . 44
4.4.3 Mobile Manipulator Trajectory Planning . . . 46
4.5 Summary . . . 48

5 MULTIPLE-CRITERIA MOTION PLANNING . . . 49
5.1 Introduction . . . 49
5.2 Background: Trajectory optimization as probabilistic inference . . . 50
5.3 Proposed method for trajectory optimization . . . 51
5.3.1 Robot trajectories as mixtures of Gaussian processes . . . 51
5.3.2 Probabilistic inference with an MGP prior . . . 53
5.3.3 Stochastic gradient estimation . . . 54
5.3.4 Updating the MGP weights during optimization . . . 56
5.3.5 MGPTO algorithm . . . 56
5.4 Implementation details of the proposed method . . . 58
5.4.1 Convergence criteria . . . 58
5.4.2 Torque optimization . . . 58
5.4.3 Geometry-aware singularity avoidance . . . 59
5.4.4 Joint limits . . . 59
5.5 Experimental results . . . 60
5.5.1 Box obstacle benchmark . . . 61
5.5.2 Robot arm in a complex environment benchmark . . . 62
5.5.3 Torque optimization benchmark . . . 64
5.5.4 Singularity avoidance experiments . . . 66
5.5.5 Mobile manipulator experiment . . . 68
5.5.6 Dual-arm robot real-world experiment . . . 70
5.5.7 Discussion . . . 71
5.6 Summary . . . 73

6 HUMAN-AWARE MOTION PLANNING . . . 74
6.1 Introduction . . . 74
6.2 Related work . . . 75
6.2.1 Perception and motion planning in dynamic environments . . . 76
6.2.2 Human motion prediction . . . 78
6.3 Proposed framework overview . . . 79
6.4 Human motion dataset . . . 81
6.5 Image processing . . . 83
6.5.1 Image segmentation . . . 83
6.6 Mapping - predicted composite distance fields . . . 84

6.7 Human trajectory prediction . . . 86
6.7.1 Intention recognition . . . 86
6.7.2 Trajectory optimization . . . 90
6.7.3 Evaluation . . . 92
6.8 Receding horizon and predictive GP motion planner . . . 97
6.9 Live hardware experiments . . . 98
6.9.1 *Change of places* - prediction is needed . . . 100
6.9.2 *Change of places with obstacle* - limitations of the CVM . . . 100
6.9.3 *Multi-goal* - robust to intention recognition . . . 100
6.9.4 Additional capabilities . . . 100
6.10 Simultaneous motion planning and trajectory prediction . . . 103
6.10.1 Joint factor graph . . . 104
6.10.2 Iterative incremental inference . . . 105
6.10.3 Evaluation . . . 106
6.11 Additional remarks . . . 108
6.12 Summary . . . 110

7 CONCLUSIONS AND OUTLOOK . . . 111

BIBLIOGRAPHY . . . 115

CURRICULUM VITAE . . . 132

FULL LIST OF PUBLICATIONS . . . 133

ŽIVOTOPIS . . . 135

1

Introduction

The introduction chapter explores the motivation and rationale behind the research conducted within the scope of this thesis. First we shall discuss what exactly the robot motion planning problem is and why it has been a paramount research topic in robotics for the past couple decades. We shall then elucidate why high-dimensional problems in particular currently present an important research challenge in this field. Subsequently, we make an argument for trajectory optimization as a motion planning approach that successfully tackles high-dimensional problems and leads to safe and efficient operation of complex robotic systems in real-world, dynamic environments with human presence. Furthermore, we elaborate why continuous-time Gaussian processes (GPs) were utilized as robot trajectory representations throughout this thesis. Thereafter, we present the original scientific contributions of this thesis and minutely describe them, while also providing the outline and structure of the thesis along with a brief summary of each chapter.

1.1 Motivation and problem statement

Motion planning is one of the fundamental problems in robotics. Successful motion planning is indispensable for achieving safe and efficient robot operation and interaction in real world environments, be it a mobile robot navigating in planar environments, a robotic arm manipulating objects or a more complex robotic systems that aspires to do both. Motion planning methods attempt to generate robot trajectories that are feasible and reach the goal optimally according to a given criterion that may vary depending on the task, robot or the environment it operates in. Generally, feasibility is congruous to a robot satisfying constraints such as being collision-free with the environment while reaching the desired goal and optimality corresponds to cost efficiency, often implying trajectory smoothness, minimization of acceleration, executed path lengths or a similar qualitative metric. Computational efficiency is also particularly important, since trajectory replanning might be needed in constantly changing and dynamic environments.

In recent years, we have seen tremendous advances in both research and application of motion planning methods for robots of up to 6 degrees of freedom (DoF), e.g., mobile robots transporting goods in a warehouse, industrial robotic arms executing tasks like cutting and welding, or unmanned aerial vehicles (UAVs) in reconnaissance missions. While robots are becoming ubiquitous for such clearly defined tasks that require zero or little interaction with humans, we may ask ourselves why there are no robots helping us in everyday tasks in our

homes or workplaces. Where is the robot that will assemble a piece of furniture or wash the dishes? The short answer is, for complex tasks, we need complex robotic systems. Moreover, even for tasks that are simple to humans, we need complex robotic systems. A human body has more than 200 DoF. If we strive for robots that can replicate certain aspects of human behavior, we need more complex robots and, correspondingly, we need methods that are able to provide good motion plans for such robots. A service robot that has to bring a cup of coffee to a person in the other room requires a mobile base for navigating between rooms, an arm for grasping the coffee cup and probably an additional arm for opening a door. Such a simple task thus requires solving a motion planning problem for a robot that has more than 10 DoF, which is often unfeasible to do in real time with the common motion planning approaches on current hardware. There is obviously a gap between robot capabilities that we envision and motion plans that state-of-the-art methods can consistently provide in real time. For the aforementioned reasons, in this thesis we opted to tackle high-dimensional problems in particular. We essentially consider motion planning for any robot that possesses more than 6 DoF as a high-dimensional motion planning problem. In particular, throughout this thesis we work with multiple redundant 7 DoF robotic arms, a 14 DoF dual-arm robot and mobile manipulators with 8 and 10 DoF.

The three prevalent ways to tackle the motion planning problem in high-dimensional spaces include grid-based methods, sampling-based methods and optimization-based methods. Grid-based methods are complete, meaning they always find a solution in finite time if a solution exists. However, these methods inherently suffer from the so-called *curse of dimensionality*. The motion planning problem becomes exponentially more difficult with each additional robot DoF, making these methods computationally intractable for high DoF systems such as humanoid robots or mobile manipulators. Sampling-based methods are probabilistically complete, meaning that the probability they will produce a solution approaches one as more time is spent. Their main disadvantage is that the obtained paths often exhibit jerky and redundant motion and therefore require post-processing to smooth and shorten the computed trajectories, which extends the computational burden. While grid-based and sampling-based methods possess the ability to *always* find solutions to high-dimensional problems, they are currently unable to reliably do so in real time, especially for problems with difficult constraints or in dynamic environments. Recently, trajectory optimization methods have become a subject of increased interest. These methods are computationally efficient for high-dimensional problems and their execution time does not scale exponentially with the number of robot DoF. However, they only find a locally optimal solution, which implies that trajectory optimization methods sometimes fail to find a valid, feasible motion plan, even if one exists. In this thesis, we opt for the trajectory optimization approach to high-dimensional motion planning because we believe that alleviating their local minima problem enough to reliably work in practical, real-world environments presents an easier challenge than fixing the inherent disadvantages of grid-based and sampling-based methods. Trajectory optimization methods scale well to high-dimensional spaces, produce smooth motion plans and can take into account many different task-related cost functions and constraints. Due to their computational efficiency, they are suitable for operation in dynamic environments.

Trajectory optimization methods usually rely on discrete-time trajectory representations,

i.e., they compute trajectories that are represented as sets of robot states at discrete time instants. However, if a trajectory is parameterized with a small number of discrete states, thin obstacles that reside between the robot positions at two adjacent states may be overlooked which would result in a collision. Moreover, a sparse solution trajectory might require post-processing for execution on the robot. On the other hand, if a trajectory is densely discretized, the optimization might have an unnecessarily high computational cost. For this reason, throughout this thesis we employ continuous-time trajectory representations, specifically, we represent trajectories as continuous-time GPs. GPs are a mathematically well-grounded means to represent continuous trajectories with a small number of discrete states and then use efficient interpolation to query trajectory states at any time of interest, allowing us to circumvent the drawbacks of discrete time representations. They allow us to use a sparsely parameterized trajectory in optimization, leading to low computational cost, while also enabling dense collision checking and efficient post-processing, if required.

1.2 ORIGINAL CONTRIBUTIONS

The original contributions of the thesis provide trajectory optimization methods for high-dimensional robot motion planning. Each devised method revolves around a particular challenge of existing trajectory optimization methods, such as alleviating the local minima problem, enabling inclusion of multiple optimization criteria and enabling operation in real-world dynamic environments. We introduce the three contributions and a brief elaboration of each of them:

#1 A method for stochastic high-dimensional robot trajectory optimization based on cross-entropy and heteroscedastic Gaussian processes.

The first contribution covers a novel trajectory planning method relying on stochastic optimization in order to find a collision-free trajectory generated from a continuous-time GP. The introduced heteroscedasticity of the underlying GP, coupled with a sparsity-exploiting covariance estimation, enables generating motion priors better suited for collision avoidance. The proposed cross-entropy based stochastic optimization for importance sampling-based trajectory optimization yields a more thorough exploration of the solution space in complex environments than state-of-the-art methods.

#2 A high-dimensional trajectory optimization method based on stochastic gradient estimation using mixtures of Gaussian processes that enables inclusion of multiple complex criteria.

The second contribution comprises a novel representation of trajectories as samples from a mixture of continuous-time GPs and an associated stochastic gradient estimation method that updates the mixture parameters in a cost-minimizing manner. The advantages stemming from the proposed method include the ability to successfully tackle high-dimensional motion planning problems while retaining trajectory smoothness, being able to find multiple trajectory solutions and supporting the inclusion of non-differentiable cost functions.

#3 An incremental factor graph inference method for simultaneous moving object tracking and robot motion planning based on Gaussian processes.

The third contribution encompasses an algorithm for simultaneous robot motion planning and moving object tracking and trajectory prediction, a context-aware algorithm for human trajectory prediction and a practical perception and motion planning pipeline that enables integration of said algorithms in a complete framework capable of predictive and pre-emptive motion planning in real-world dynamic environments. The resultant framework enables computing whole-body movements of a mobile manipulator that account for the predicted motion of moving obstacles, including humans.

1.3 OUTLINE OF THE THESIS

This thesis is organized into seven chapters. Each chapter begins with a brief abstract which aims to concisely present the method proposed in the chapter and its contributions. Afterwards, the reader is gradually introduced to the problem and motivation behind the proposed solution. In the end of each chapter, a short summary is given which restates some of the main results of the chapter, obtained experimental results and conclusions that stem from them. Hereafter, we present the outline of this thesis with a brief summary of each chapter's contents.

Ch 2 This chapter provides an overview of existing methods for robot motion planning in high-dimensional spaces. First, the motion planning problem formulation is presented and some of the early works in the field are described. Subsequently, the chapter provides a structured insight into the three main ways to tackle the motion planning problem, namely grid-based approaches, sampling-based approaches and trajectory optimization approaches. Since each method proposed in this thesis belongs to trajectory optimization approaches, these are further divided into gradient-based and gradient-free methods with the relevant work in the field emphasized and minutely described.

Ch 3 This chapter presents the general theoretical background and explains common concepts that are relevant throughout this thesis. First, we present the continuous-time GP trajectory representations with a special prior suitable for robot motion planning, which form the backbone for each proposed contribution. Second, we describe the relied-upon robot model and environment representation used for collision checking in all experiments conducted in this thesis. In the end, we introduce the problem of kinematic singularities and describe a novel geometry-aware singularity index that is later utilized by one of the proposed methods for singularity identification and avoidance within trajectory optimization.

Ch 4 In this chapter we present a novel method for high-dimensional robot motion planning in cluttered environments. Firstly, the heteroscedastic continuous-time GP prior is presented and its benefits regarding improved performance in collision avoidance in complex environments are discussed. This is followed by the derivation of a

cross-entropy based stochastic trajectory optimization method that enables a more thorough exploration of the solution space than the current state of the art, while having comparable execution time. The proposed method is then evaluated in three simulated scenarios, which aim to demonstrate the improved performance in collision avoidance and the ability to efficiently tackle high-dimensional trajectory planning problems.

Ch 5 A novel multi-criteria high-dimensional motion planning method based on mixtures of Gaussian processes (MGPs) and stochastic gradient estimation is presented. In the beginning, the MGP trajectory representation is proposed with discussion about the advantages it imposes, such as the ability to generate multiple solution trajectories and enabling better exploration of the trajectory space than the GP representation. Thereafter, the stochastic gradient estimation procedure that enables inclusion of non-differentiable optimizing costs is presented, followed by a meticulous experimental evaluation that features multiple simulation benchmarks with different robots and real-world experiments. In the end, a discussion about the obtained results is provided, offering insights about the achieved performance in collision avoidance, torque minimization and singularity avoidance while retaining trajectory smoothness and computational efficiency.

Ch 6 This chapter presents a novel complete perception and motion planning framework that accounts for the predicted motion of humans, enabling predictive motion planning in real-world dynamic environments. First an overview of related work in robot perception for dynamic environments and human trajectory prediction is given, followed by the proposed framework outline. Then, the novel human motion dataset is described. Subsequently, each framework component is presented, describing the practical real-time mapping approach, the novel human trajectory prediction method and the motion planning method based on trajectory optimization capable of real-time operation. The resultant framework is then validated in multiple real-world experiments with a mobile manipulator, showcasing the ability to support predictive whole-body motion planning in environments with human presence, while relying on a single onboard RGB-D sensor data humans. In the end, an alternative trajectory optimization formulation was given, proposing simultaneous motion planning and human trajectory prediction.

Ch 7 In this chapter we conclude the thesis and give a summary of each presented scientific contribution. We also offer some ideas for future work in the field of trajectory optimization for high-dimensional motion planning.

2

An overview of motion planning methods

This chapter presents provides an overview of existing methods for robot motion planning in high-dimensional spaces. We first introduce the reader to the problem of motion planning, mentioning some of the early works in the field and presenting the three dominant ways to tackle it, namely grid-based approaches, sampling-based approaches and trajectory optimization approaches. Then we explore each of the mentioned approaches, discussing their advantages and appropriate use-cases, while also describing the most influential methods in each subfield. We further divide trajectory optimization approaches into gradient-based and gradient-free, presenting their inherent differences and discussing related methods in more detail, as each method proposed and implemented in the scope of this thesis relies on trajectory optimization.

2.1 Introduction

A fundamental need in robotics is to have algorithms that convert high-level specifications of tasks into low level descriptions of how the robot should move. The term motion planning is often used for these kinds of problems [1]. Motion planning is therefore an indispensable tool for robots that aspire to navigate through an environment without collisions. Motion planning algorithms attempt to generate trajectories through the robot's configuration space that are both feasible and optimal based on some performance criterion that may vary depending on the task, robot or environment. Generally, feasibility of a trajectory often refers to collision avoidance and constraints assurance, such as robot joint limits, while optimality often corresponds to trajectory smoothness, minimization of acceleration, executed path length, or a similar qualitative metric. There are two common paradigms for successful robot motion planning. The first paradigm separates motion planning into path planning and path execution. The second paradigm directly finds the trajectory of the robot - it takes into account the robot kinematics or dynamics to plan the full robot motion from one point to the other. Path planning is used to find the shortest, or otherwise optimal, collision-free path between two points in environment while not taking into account the temporal component of robot motion. Path execution is then used to follow the planned path, leveraging methods from control theory. The initial attempts to solve path planning problems included grid search methods such as the A* algorithm [2], reactive planners such as bug algorithms [3], combinatorial planning methods originating from computational geometry such as cell decomposition [4], visibility graphs [5] and Voronoi diagrams [6, 7].

All of the mentioned methods are complete, meaning that they find a solution if it exists and report failure otherwise. They present elegant solutions for low dimensional configuration spaces in environments that feature static obstacles. However, bug algorithms produce unnecessarily long paths, while grid search methods and combinatorial approaches suffer from the so-called curse of dimensionality, i.e., they quickly become computationally intractable with the increase of the configuration space dimension. The curse of dimensionality implies that fast replanning is not possible in high-dimensional problems, making those approaches ineffective for complex robots, e.g., mobile manipulators, in dynamic environments.

The increasing complexity of robots and the environments that they operate in has spurred the need for effective high-dimensional motion planning. Consider, for instance, a personal robot operating in a cluttered household environment or a humanoid robot performing navigation and manipulation tasks in an unstructured environment. Efficient motion planning is important to enable these high DoF robots to perform useful tasks, subject to motion constraints while avoiding collisions with obstacles in the environment. Processing time is especially important in dynamic environments where replanning is necessary. These considerations lead to the development of grid-based methods with ameliorated efficiency and to the development of sampling-based motion planning methods which offer weaker guarantees than combinatorial methods but are more efficient. Sampling-based methods have become popular in the domain of high-dimensional motion planning, including planning for manipulation with robotic arms. They abandon the concept of explicitly characterizing the configuration space and build upon the notion of connecting randomly sampled points from its free parts – they check the randomly drawn samples for collisions in the environment to figure out if a sample configuration is a part of the free configuration space or not. Sampling-based approaches often provide formal guarantees on finding the solution and most of them exhibit probabilistic completeness, i.e., the probability of generating a solution approaches one as more time is spent [8].

All of the discussed techniques so far aim at capturing the connectivity of free configuration space into a graph. The exigency for fast and efficient planning methods lead to development of the planning paradigm which directly finds the trajectory of the robot, accounting for the temporal component of robot motion. Potential field methods were the initial approach of that paradigm. They model the robot, which is represented as a point in configuration space, as a particle under the influence of an artificial potential field that is formed by the attractive force from goal point and repulsive forces from obstacles in configuration space. To obtain the policy which moves the robot safely towards the goal, one would simply perform gradient descent on the potential function. Main disadvantages of potential field methods are the lack of completeness and optimality guarantees and the presence of local minima for a non point-mass robot.

Recently, the trajectory optimization methods that are appropriate for very high DoF robots have been proposed. In these methods the robot trajectory is encoded as a sequence of states and controls, which offers several advantages for robot motion planning, e.g., they inherently consider the robot kinematics. Furthermore, the obtained trajectories can often be executed directly by the robot without the need for post-processing, contrary to paths generated by grid-based and sampling-based methods. Trajectory optimization approaches

start with an initial trajectory and then optimize it according to an arbitrary state-dependent objective function that in most cases encodes a sum of the smoothness cost, the collision-avoidance cost and possibly other task-dependent costs and constraints. While trajectory optimization methods are computationally efficient even for very high-dimensional spaces, their main downside is the lack of formal guarantees, leading to failure cases where they converge to infeasible local minima which is especially the case in cluttered environments with plethora of obstacles.

In this chapter, we will explore the three aforementioned approaches to motion planning and the representative methods for each one. We pay special attention to trajectory optimization methods as the high-dimensional motion planning methods proposed in this thesis build upon them. First, in Section 2.2 we elaborate grid-based methods to motion planning, followed by discussing sampling-based planning methods in Section 2.3. In the end, in Section 2.4 we explore the body of work in the trajectory optimization subfield, which we additionally divide into gradient-based and gradient-free methods.

2.2 GRID-BASED APPROACHES

Grid-based motion planning approaches overlay a discrete grid over the configuration space and assume each grid point corresponds to a particular configuration. The robot is allowed to move to adjacent grid points as long as the line between them is collision-free, i.e., contained within free configuration space. This discretizes the set of actions, and search algorithms are used to find a path from the start to the goal. The pioneering work in the domain of grid-based motion planning is the A* search algorithm [2] that employs a heuristic function in order to find the shortest path between the start and the goal state. The D* algorithm [9], along with its improved variants Focussed D* [10] and D* Lite [11], resembles A* and is capable of planning paths in unknown, partially known, and changing environments in an optimal and complete manner. Since these classical grid-based methods are complete, they are still commonly used for robot motion planning in environments featuring bottlenecks and other such narrow passages, although in most cases for low dimensional configuration spaces, in particular for mobile robots navigating in the 2D plane.

While naive application of A* is typically unsuitable for high-dimensional planning problems, since number of points on the grid grows exponentially in the configuration space dimension, current research has made advances in applying grid search to systems with many DoF. Until recently, a major problem with A* and related algorithms has been that admissible heuristics result in examination of prohibitively large portions of the configuration space, whereas inflated heuristics cause significantly suboptimal behavior. In [12], a framework for efficiently updating the A* search while smoothly reducing heuristic inflation is presented, allowing resolution complete search in an anytime fashion on a broader variety of problems than previously computable. It starts by finding a suboptimal solution quickly using a loose bound, then tightens the bound progressively as time allows. Given enough time it finds a provably optimal solution. Additional work has examined inducing smoother motion while reducing the cardinality of the action set [13], as well as reusing partial plans discovered by past searches [14]. The A*-Connect method [15] uses a fast approximation of the front-to-front heuristic to lead the forward and backward searches towards each other, while

retaining theoretical guarantees on completeness and demonstrating its performance in manipulation planning.

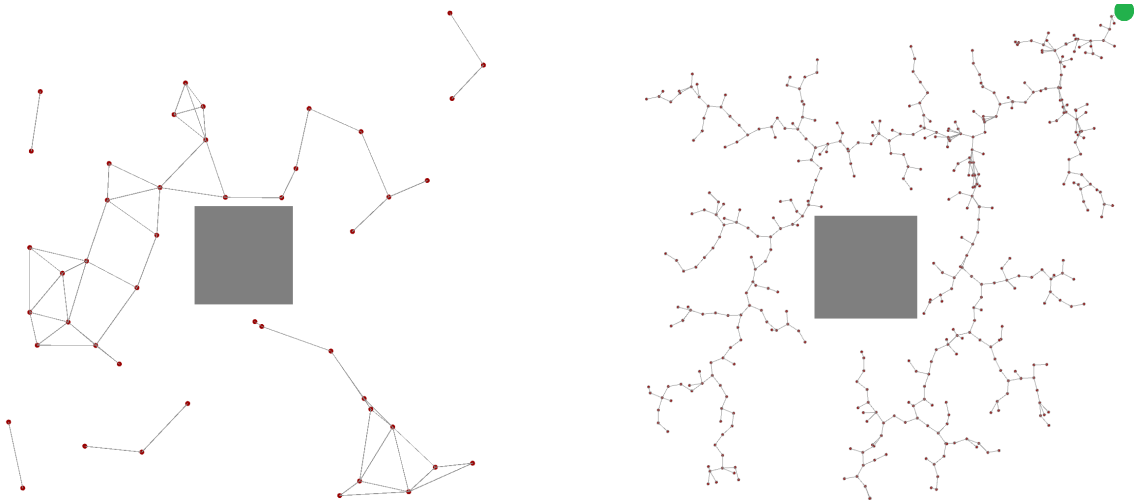
All of the so far discussed methods require setting a fixed grid resolution. While search is faster with lower resolution, grid-based methods will fail to find paths through narrow portions of free configuration space if the resolution is too coarse. To achieve a better balance between computational efficiency and the ability to find collision-free paths, a multi-resolution A* algorithm has been proposed [16], using a fine discretization in areas with obstacles, and a coarse discretization in free areas of the configuration space. The mentioned method demonstrated competitive performance in manipulation planning for a 7 DoF robot arm. Recently, there have been attempts to parallelize the grid search [17, 18], providing a speed-up that is almost linear in the number of threads used for parallelization. However, while the computational time significantly lowers by exploiting multithreading, the grid search still requires significant hardware resources.

While grid-based methods offer guarantees on completeness, their main drawback is computational complexity, as even the state-of-the-art methods often suffer from the curse of dimensionality, and become computationally intractable for very high DoF systems such as humanoid robots. Despite the recent improvements in search heuristics, exploitation of multiple resolution grids and leveraging parallelization, grid-based methods are unable to ensure efficient and reliable high-dimensional robot operation, especially in dynamic environments.

2.3 SAMPLING-BASED METHODS

Sampling-based approaches have become popular in the domain of high-dimensional motion planning, including manipulation planning. In a sense, these approaches attempt to capture the connectivity of the robot configuration space by sampling it [19]. Randomized approaches have its advantages in terms of providing efficient solutions for challenging problems [1]. The downside is that the solutions are widely regarded as suboptimal. Sampling-based planners are understood to be probabilistically complete [20], a weaker notion of completeness that ensures a solution will be provided, if one exists, given sufficient runtime of the algorithm (even if it means infinite runtime). However, unlike grid-based methods, they cannot determine if no solution exists.

Sampling-based methods typically work in a two-step fashion. First, a collision-free path is discovered without regard for any measure of cost. Second, the obtained path is improved by applying certain heuristics. For the first step, a Monte-Carlo algorithm for path planning with many DoF [21] was a seminal work that demonstrated a solver for impressively difficult problems. Perhaps the two most commonly used algorithms are the probabilistic roadmap (PRM) [22] and rapidly exploring random trees (RRT) [23]. The PRM method was shown to be well suited for path planning in configuration spaces with many DoF, and with complex constraints, including kinodynamic [24, 25, 26]. RRT has also been applied to contend with differential constraints, and was shown to be successful for general high dimensional planning [27]. The intuitive implementation of both RRT and PRM and the quality of obtained solutions lead to their widespread adoption in robotics and many other fields. Even though the idea of connecting points sampled randomly from



(a) (b)

Figure 2.1: Left: roadmap built in the PRM learning phase [19]. Right: RRT exploring environment with one obstacle after 500 iterations [19]. The root of the tree is shown as green circle.

the state space is essential in both RRT and PRM, these two algorithms differ in the way that they construct a graph connecting these points [8].

PRM implements two main procedures to generate a probabilistic roadmap. A learning phase occurs first, where the configuration space is randomly sampled for a certain amount of time. The sampled configurations are declared as vertices if they lie in free configuration space and are connected to nearby vertices with a local planner, while those in the obstacle space are discarded. This is followed by a query phase where the start and goal configurations are defined and connected to the roadmap. A graph search is then performed to find the shortest path through the roadmap between start and goal configurations. An example of a roadmap built in the PRM learning phase is depicted in Fig. 2.1. Roadmaps are sometimes referred to as forests, as an analogy to trees in RRT. PRM has the inherent ability to solve different instances of the planning problem in the same environment, which is a product of maintaining the roadmap and specifying start and goal configurations in a subsequent stage. It is therefore referred to as a multi-query planner. Planning time is invested in sampling and generating a roadmap, but queries are solved quickly. PRM was initially developed for articulated robots, but has been extended for non-holonomic car-like robots. Even though sampling-based methods do not explicitly optimize a cost function, authors in [8] proposed the PRM* algorithm, which is an adaptation of PRM that was proven to be asymptotically optimal, alleviating one of the main downsides of the original PRM method.

RRT represents another category of sampling-based planners, which are single-query planners. A tree is incrementally grown from the start configuration to the goal configuration, or vice versa. A configuration is randomly selected in the configuration space. If it lies in the free space, a connection is attempted to the nearest vertex in the tree. As a result of uniform sampling, RRT is more likely to select samples in larger Voronoi regions and the tree is incrementally and rapidly grown towards that free space [19]. An example of a tree generated by RRT algorithm exploring environment with one obstacle is shown in Fig. 2.1. For single query problems, RRT is faster compared to PRM, since it does not require a learning phase.

Similarly to adapting PRM to have a notion of optimality, the authors in [8] also proposed the RRT* algorithm, proving its asymptotical optimality. Even though it still suffers from the curse of dimensionality, RRT* is perhaps the most commonly used sampling-based planner in high-dimensional configuration spaces, with many developed extensions [28] that try to improve its performance by relying on heuristics that *inform* the search [29] or consider kinematic and dynamic constraints [30, 31] to facilitate its applications for complex real-world robots.

Other sampling-based path planners of note include expansive space trees (EST) [32] and sampling-based roadmap of trees (SRT) [33]. The EST employs a function that sets the probability of node selection based on neighboring nodes, unlike RRT where sampling is uniform. The SRT combines the main features of multiple-query algorithms such as PRM with those of single-query algorithms such as EST and RRT. For improving the planned path, one popular approach is the *shortcut* heuristic, which picks pairs of configurations along the path and invokes a local planner to attempt to replace the intervening sub-path with a shorter one [34, 35]. Methods such as medial axis and partial shortcuts have also proven effective [36].

As the sampling-based planners became increasingly well understood in recent years, it was suggested that randomization may not, by itself, account for their efficiency [37]. It was shown that quasi-random sampling sequences can accomplish similar or better performance than their randomized counterparts [38]. Nevertheless, many sampling-based methods are computationally inefficient for challenging high-dimensional problems, especially if the environment is dynamic and replanning is needed. They may suffer from the curse of dimensionality and frequently spend sizable computational effort for sampling parts of the configuration space that might be irrelevant for the task. Furthermore, sampling-based methods rarely take into account trajectory smoothness and consequently the obtained paths necessitate post-processing to avoid jerky or redundant motion. To mitigate the mentioned deficiencies, methods combining potential fields with sampling based planning [39, 40] have been introduced. However, they are unable to optimize secondary costs (e.g. motor torque minimization) and for high-dimensional spaces they either rely on human demonstrations or still require significant computational effort.

2.4 Trajectory optimization

The key motivation for trajectory optimization is the focus on producing *optimal motion*: incorporating dynamics, smoothness, obstacle avoidance, joints limits and any other arbitrary task-dependent constraints in a mathematically precise objective. Despite a rich theoretical history and successful applications, most notably in the control of spacecraft and rockets, trajectory optimization techniques have had limited success in robot motion planning. Much of this may be attributed to two causes; the large computational cost for evaluating objective functions and their higher-order derivatives in high-dimensional spaces, and the presence of local minima when considering motion planning as a generally non-convex continuous optimization problem.

Early works in this area proposed the use of potential fields for avoiding both static and dynamic obstacles. The method's sensitivity to local minima has been addressed in

a range of related work. Analytical navigation functions that are free of local minima for some specific environments were proposed in [41]. A global potential field to push the robot away from configuration space obstacles, starting with a trajectory that was in collision was proposed in [42]. In [43], the free part of configuration space can is locally approximated as a union of spheres around the current trajectory as a part of a local optimization that tries to shorten and smooth the trajectory. The trajectory is modelled as a mass-spring system, an elastic band, and replanning is performed by scanning back and forth along the elastic, while moving one mass particle at a time. The cost function contains terms that attempt to control the motion of the trajectory particles along the elastic. However, significant computational effort is required to construct a model of the free space. This method was further extended in [44] where the real-time application to a high DoF humanoid robot was presented. The aforementioned approaches locally approximate the free space using a union of spheres, which is an overly conservative approximation and may not find feasible trajectories even if they exist.

More recent trajectory optimization methods play two important roles in robot motion planning. First, they can be used to smooth and shorten trajectories computed by other planning methods such as sampling-based planners. Second, they can be used to compute collision-free trajectories from scratch by starting with naive trajectory initializations that might be in collision with obstacles [45], and then minimize an appropriate objective function encoding collision avoidance and other relevant costs. The state-of-the-art trajectory optimization methods all start with an initial trajectory, most commonly a constant-velocity straight line, and then minimize an objective function in order to optimize the trajectory according to some criterion. The goal of trajectory optimization for motion planning is to obtain trajectories $\boldsymbol{\theta}(t)$ that minimize costs and satisfy constraints. Motion planning can thus be formalized as

$$\begin{aligned} \text{minimize} \quad & \mathcal{F}[\boldsymbol{\theta}(t)] \\ \text{subject to} \quad & \mathcal{G}_i[\boldsymbol{\theta}(t)] \leq 0, i = 1, \ldots, m_{ineq} \\ & \mathcal{H}_i[\boldsymbol{\theta}(t)] = 0, i = 1, \ldots, m_{eq} \end{aligned} \tag{2.1}$$

where the trajectory $\boldsymbol{\theta}(t)$ is a continuous-time function that maps time t to robot states, which are usually robot configurations and sometimes higher order derivatives. $\mathcal{F}[\boldsymbol{\theta}(t)]$ is cost functional or objective that encodes collision costs that enforce the trajectory to be collision-free and the quality of a trajectory that usually corresponds to smoothness and is evaluated as a minimization of velocity or acceleration [46]. $\mathcal{G}_i[\boldsymbol{\theta}(t)]$ represents inequality constraint, for example the joint angle limits, while $\mathcal{H}_i[\theta(t)]$ represents task-dependent equality constraints, such as the desired start and end configurations and velocities, or the desired end-effector orientation (for example, holding a cup of coffee upright). Depending on the specific problem, the number of inequality constraints may be zero. Collision cost may also appear as an obstacle avoidance inequality constraint [45], depending on the optimization technique used to solve (2.1). Most existing trajectory optimization algorithms in practice work with a finely discretized trajectory, which is suitable for reasoning about thin obstacles or tight navigation constraints, but can cause a large computational cost.

2.4.1 *Gradient-based methods*

A seminal work in the modern gradient-based trajectory optimization is the covariant Hamiltonian optimization for motion planning (CHOMP) [47, 48]. CHOMP revived interest in trajectory optimization by demonstrating the effectiveness in motion planning on several high DoF robotic platforms, including a mobile manipulation platform and a quadruped. The key feature of CHOMP is a formulation of trajectory costs that are invariant to the time parameterization of the trajectory. In order to produce smooth robot motion while avoiding obstacles, CHOMP uses two objective functionals: a smoothness term which captures dynamics of the trajectory and an obstacle functional which captures the requirement of avoiding obstacles and preferring margin from them. They define the smoothness functional in terms of a metric in the space of trajectories. The obstacle functional is developed as a line integral of a scalar cost field c - a precomputed signed distance field, defined so that it is invariant to retiming. Regardless of how fast or slow the arm moves through the field, it will accumulate the exact same cost. The two functionals have complementary role: the obstacle functional governs the shape of the path, and the smoothness functional governs the timing along the path. Several augmentations of CHOMP have been proposed to improve its performance in specific areas. Multigrid CHOMP with local smoothing [49] improves the runtime of CHOMP under constraints, without significantly reducing optimality. T-CHOMP [50] is a functional gradient algorithm that directly optimizes in space-time, thus being able to successfully incorporate constraints and cost functions that explicitly depend on time. Incremental trajectory optimization algorithm (ITOMP) [51] takes into account moving obstacle and thus enables real-time replanning in dynamic environments. CHOMP was applied in many use cases, such as collaborative manipulation when coupled with a human motion prediction method [52], dual-arm assembly of ring-shaped elastic objects [53] and unmanned aerial vehicles (UAV) planning in dynamic environments [54].

An important shortcoming of CHOMP and related methods is the need for many trajectory states in order to reason about fine resolution obstacle representations or find feasible solutions when there are many constraints. The framework called TrajOpt [45, 55] thus formulates motion planning as sequential quadratic programming problem and features convex collision checking. Sequential convex optimization involves solving a series of convex optimization problems that approximate the cost and constraints of the original problem. The ability to add new constraints and costs to the optimization problem allows TrajOpt to tackle a larger range of motion planning problems, including planning for underactuated, non-holonomic systems. For collisions, TrajOpt uses signed distances using convex-convex collision detection and takes into account continuous-time safety by considering the swept-out volume of the robot between time steps. This formulation allows for sparsely sampled trajectory, which makes TrajOpt very computationally efficient in practice. However, if the smoothness is required in the output trajectory, either a densely parameterized trajectory or post-processing of the trajectory might still be needed thus increasing computation time.

A continuous-time trajectory representation can avoid the computational cost incurred by using large number of states and yield a more efficient approach, while maintaining the smoothness of trajectory. Linear interpolation, splines, and hierarchical wavelets have been used to represent trajectories in filtering and state estimation. Recently, B-splines [56] and

kernel methods [57] have similarly been used to represent trajectories with fewer states in motion planning problems.

The Gaussian process motion planning family of algorithms [58, 59, 60, 46, 61, 62] uses continuous-time trajectory representation; specifically, they view trajectories as functions that map time to robot states. The continuous-time trajectory is represented as a sample from a GP generated by a linear time-varying stochastic differential equation. They show that GPs inherently provide a notion of trajectory optimality through a carefully-selected motion prior. Efficient structure-exploiting GP regression can be used to query the trajectory at any time of interest in $\mathcal{O}(1)$ complexity [63]. Using this representation, they developed a gradient-based optimization algorithm called GPMP that can efficiently overcome the large computational costs of fine discretization while still maintaining smoothness in the result. Through the GP formulation, they also cast the motion planning as probabilistic inference [64, 65]. Similar to how the notion of trajectory optimality is captured by a prior on trajectories, the notion of feasibility can also be viewed probabilistically as well and encoded in a likelihood function. Bayesian inference can then be used to compute a solution to the motion planning problem efficiently through the use of factor graphs [66]. The duality between inference and optimization allows performing efficient inference on factor graphs by solving sparse least squares problems, thereby exploiting the structure of the underlying system. Similar techniques have been used to solve large-scale Simultaneous Localization and Mapping (SLAM) problems [67]. With this key insight, preexisting efficient optimization tools developed by the SLAM community can be used in the context of motion planning. This results in the GPMP2 algorithm, which is more efficient than previous motion planning algorithms. A useful property of GPMP2 is its extensibility and applicability for wide range of problems. For example, combined learning from demonstration and motion planning [68] presented an efficient approach to skill learning and generalizable skill reproduction. In [69], a framework for avoidance of singular robot configurations and manipulability maximization was proposed, while a unified probabilistic framework for trajectory estimation and planning was provided in [70]. The main drawback of GPMP algorithms is that they are limited in their ability to handle motion constraints like nonlinear inequality constraints.

2.4.2 *Stochastic optimization methods*

All of the methods mentioned in the previous section rely on first order information about the cost functionals in order to carry the optimization. In some use cases, the use of gradient-based optimization might not be the best way solve the problem. Notably, in particularly cluttered environments there exists a plethora of local minima in which gradient-based optimization method may converge, leading to infeasible trajectories and making gradient-based methods unsuitable for reliable motion planning. The local minima problem happens due to the fact that it is possible for a trajectory that is in collision briefly but generally stays far away from obstacles to have lower cost than a trajectory that never collides. Moreover, depending on the task we could potentially have non-differentiable constraints or cost functions that do not have available derivations.

In such cases, stochastic optimization techniques might be more suitable to foster

the avoidance of local minima and optimize non-differentiable cost functions, although in most cases they are more computationally demanding than gradient-based methods. Stochastic trajectory optimization methods combine sampling and trajectory optimization to overcome the local minima problem present in gradient-based optimization. Contrary to sampling-based approaches, which sample in the configuration space, stochastic trajectory optimization methods sample in the trajectory space, which may be more sample-efficient. A pioneering work in the area of stochastic trajectory optimization for motion planning was STOMP, a method that inherits CHOMP's motion planning formulation but carries out gradient-free optimization. Stochastic trajectory optimization for motion planning (STOMP) [71] explores the space around an initial trajectory by taking a batch of noisy trajectories, evaluating their cost, and calculating a stochastic gradient estimate to update the initial trajectory in a cost-minimizing manner. The key trait of STOMP is its ability to optimize non-differentiable constraints, while it also showcased improved success rate in comparison to CHOMP in complex environments. Due to its capabilities and relatively easy implementation, STOMP was extended for multicriteria optimization [72], energy-efficient planning in marine applications [73] and for supporting Cartesian path constraints [74].

Besides STOMP, multiple genetic algorithm applications to robot motion planning have been proposed [75, 76], although none showed viable motion planning performance for high DoF robots. A particle swarm filter method for trajectory optimization [77] displayed being less prone to local minima than gradient-based methods, but it did not showcase its performance in highly cluttered environments. To successfully find multiple solution trajectories, the stochastic multimodal trajectory optimization algorithm (SMTO) [78] utilized variational Bayesian expectation maximization, but it is orders of magnitude computationally slower than STOMP. Another group of methods uses cross-entropy optimization [79, 80]. In [79], the authors rely on drawing trajectories from a set of feasible paths which are then used to optimize a trajectory smoothness objective. The proposed framework generates smooth trajectories; however, sampling whole collision-free trajectories can become infeasible in complex environments.

2.5 SUMMARY

Safely navigating through an environment is one of the key tasks which an autonomous robot has to accomplish. Motion planning is an essential tool used to find trajectories of robot states that achieve a desired task. In this chapter, we presented an overview of representative methods for motion planning in high-dimensional robot configuration spaces. We first presented grid-based methods which are resolution complete and often offer optimal solutions. Their main downside is that the number of grid points grows exponentially in the configuration space dimension, which makes even state-of-the-art methods inappropriate for very high-dimensional problems. Subsequently, we introduced sampling-based methods which are efficient in most practical problems, but offer weaker guarantees than grid-based methods. Sampling-based methods rely on the concept of connecting randomly sampled points from free parts of the configuration space, i.e., they check the randomly drawn samples for collisions in the environment and connect them into an appropriate structure if they are collision-free. They are probabilistically complete, however, they often require post-processing and can still be inefficient for high-dimensional motion planning problems in particularly complex environments or with challenging constraints. In the end, we presented trajectory optimization methods that circumvent the curse of dimensionality and have demonstrated the ability to solve high-dimensional motion planning problems efficiently. Gradient-based trajectory optimization methods are particularly efficient, although they are prone to getting stuck in infeasible local minima. Stochastic trajectory optimization methods alleviate the local minima problem and allow the inclusion of non-differentiable costs and constraints, but are in most cases less computationally efficient than gradient-based methods.

3

Theoretical background

Motion planning methods introduced throughout this thesis leverage several different mathematical tools and frameworks developed within the robotics community. In this chapter, a brief theoretical background on the used tools is provided. First, in Section 3.1 we lay out the fundamentals of the continuous-time GP trajectory representations that constitute a foundation for each contribution presented in this thesis. Then, in Section 3.2 we elaborate the collision avoidance approach used throughout this thesis, including the relied-upon robot model and environment representation. In the end, in Section 3.3 we describe kinematic singularities and a way to identify them with a novel geometry-aware singularity index, which is then exploited in Chapter 5 for singularity avoidance with trajectory optimization.

3.1 Gaussian processes as trajectory representations

Robot motion planning problems are commonly formulated using discrete-time trajectory representations. This may lead to neglecting thin obstacles that reside between the robot positions at two adjacent discrete states, possibly causing collision. Furthermore, a sparsely discretized trajectory often requires post-processing in order to be executed by a real robot. In this thesis we therefore represent robot trajectories as continuous-time GPs, which bypass the mentioned drawbacks of discrete time representations. GPs enable us to employ sparsely parameterized trajectories in optimization, while at the same time supporting dense collision checking and avoiding post-processing due to fast GP interpolation. In the following subsections we present continuous-time Gaussian processes (GP) suitable for motion planning. We closely follow developments from [46] as the therein proposed trajectory representation underpins the proposed thesis contributions and their benefits. For a more in-depth treatment, we refer the reader to several books on GPs and their applications in robotics [81, 82, 83].

We consider robot trajectories represented as samples from a continuous-time GP

$$\boldsymbol{\theta}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \mathcal{K}(t, t')) \tag{3.1}$$

that is parameterized with N *support states* at discrete time instants, $\boldsymbol{\theta}_i \in \mathrm{R}^D$, $i \in N$, where D is the state dimensionality. This implies that, for any collection of times $\boldsymbol{t} = \{t_0, \ldots, t_N\}$, $\boldsymbol{\mu}(t)$ is a vector-valued mean function

$$\boldsymbol{\mu} = [\boldsymbol{\mu}(t_0) \ldots \boldsymbol{\mu}(t_N)]^T, \tag{3.2}$$

while $\mathcal{K}(t, t'))$ is a matrix-valued covariance function [46], defined as

$$\mathcal{K} = [\mathcal{K}(t_i, t_j)]|_{i,j,0\leq i,j\leq N}. \tag{3.3}$$

3.1.1 *GP prior for motion planning*

A vector-valued GP provides a theoretically grounded continuous-time trajectory representation, where trajectories are deemed as functions that map time to robot states. It defines a prior on the space of robot trajectories

$$p(\boldsymbol{\theta}) \propto \exp\Big\{-\frac{1}{2}\big\|\boldsymbol{\theta}-\boldsymbol{\mu}\big\|^2_{\mathcal{K}}\Big\}, \tag{3.4}$$

where $\big\|\boldsymbol{\theta}-\boldsymbol{\mu}\big\|^2_{\mathcal{K}} = (\boldsymbol{\theta}-\boldsymbol{\mu})^T\mathcal{K}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu})$ is the Mahalanobis distance between a trajectory $\boldsymbol{\theta}$ and the underlying GP.

In this thesis, we utilize a structured kernel that belongs to a special class of GP priors and is generated by a linear time-varying stochastic differential equation (LTV-SDE)

$$\dot{\boldsymbol{\theta}}(t) = \boldsymbol{F}(t)\boldsymbol{\theta}(t) + \boldsymbol{\nu}(t) + \boldsymbol{L}(t)\boldsymbol{w}(t), \tag{3.5}$$

where $\boldsymbol{F}$ and $\boldsymbol{L}$ are system matrices and $\boldsymbol{\nu}$ is a known exogenous input. The white noise process $\boldsymbol{w}(t)$ is itself a GP with zero mean value

$$\boldsymbol{w}(t) \sim \mathcal{GP}(\mathbf{0}, \boldsymbol{Q}_c\delta(t-t')), \tag{3.6}$$

where $\boldsymbol{Q}_c(t)$ is a positive semi-definite power-spectral density matrix. A similar dynamical system has been extensively utilized to represent trajectory distributions in state estimation [63, 84], sensor calibration [85] and robot motion planning [46, 70]. The mean and covariance of the GP generated by the LTV-SDE given in (3.5) evaluate to

$$\widetilde{\boldsymbol{\mu}}(t) = \boldsymbol{\Phi}(t, t_0)\boldsymbol{\mu}_0 + \int_{t_0}^{t}\boldsymbol{\Phi}(t, s)\boldsymbol{\nu}(s)\,\mathrm{d}s, \tag{3.7}$$

$$\widetilde{\mathcal{K}}(t, t') = \boldsymbol{\Phi}(t, t_0)\mathcal{K}_0\boldsymbol{\Phi}(t', t_0)^T + \int_{\mathrm{t}_0}^{\min(t,t')}\boldsymbol{\Phi}(t, s)\boldsymbol{L}(s)\boldsymbol{Q}_c\boldsymbol{L}(s)^T\boldsymbol{\Phi}(t', s)^T\,\mathrm{d}s, \tag{3.8}$$

where $\boldsymbol{\mu}_0$ and $\mathcal{K}_0$ are the initial mean and covariance of the first state, and $\boldsymbol{\Phi}(t, s)$ is the state transition matrix.

An interesting effect of the LTV-SDE in (3.5), due to the Markov property, is that the precision matrix $\widetilde{\mathcal{K}}^{-1}$ is exactly sparse block tridiagonal [63]

$$\widetilde{\mathcal{K}}^{-1} = \widetilde{\boldsymbol{F}}^{-T}\widetilde{\boldsymbol{Q}}^{-1}\widetilde{\boldsymbol{F}}^{-1}, \tag{3.9}$$

where

$$\widetilde{\boldsymbol{F}}^{-1} = \begin{bmatrix} \mathbf{1} & 0 & \dots & 0 & 0 \\ -\boldsymbol{\Phi}(t_1, t_0) & \mathbf{1} & \dots & 0 & 0 \\ 0 & -\boldsymbol{\Phi}(t_2, t_1) & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{1} & 0 \\ 0 & 0 & \dots & -\boldsymbol{\Phi}(t_N, t_{N-1}) & \mathbf{1} \end{bmatrix} \tag{3.10}$$

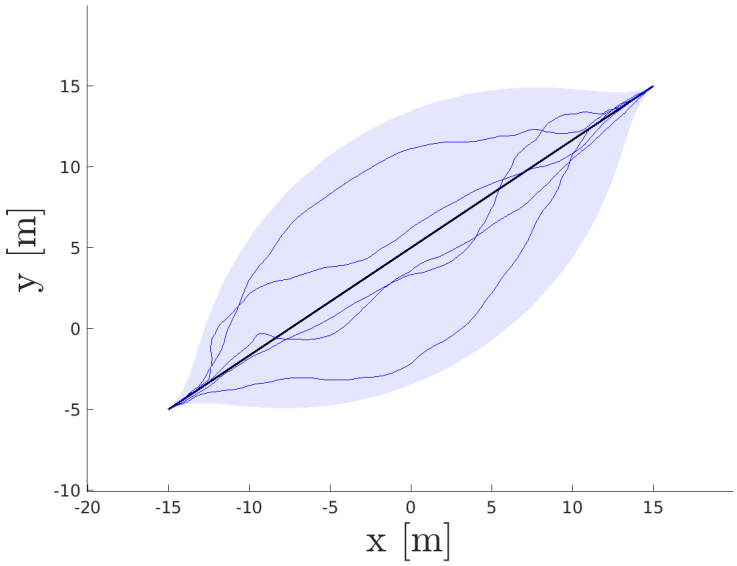


Figure 3.1: The GP prior suitable for motion planning in 2D spaces. The black line depicts its mean initialized as a straight line from start to goal position, while the blue shape represents its covariance. Blue lines illustrate examples of trajectories sampled from the pertaining GP.

with $\boldsymbol{\Phi}(t,s)$ being the state transition matrix [63], and

$$\widetilde{\boldsymbol{Q}}^{-1} = \operatorname{diag}(\boldsymbol{\mathcal{K}}_0^{-1}, \boldsymbol{Q}_{0,1}^{-1}, ..., \boldsymbol{Q}_{N-1,N}^{-1}) \tag{3.11}$$

with

$$\boldsymbol{Q}_{a,b} = \int_{t_a}^{t_b} \boldsymbol{\Phi}(t_b,s)\boldsymbol{L}(s)\boldsymbol{Q}_c\boldsymbol{L}(s)^T\boldsymbol{\Phi}(t_b,s)^T \,\mathrm{d}s. \tag{3.12}$$

Furthermore, contrary to estimation problems, in motion planning problems there exists a desired fixed goal state. Thus we condition the pertaining GP with a fictitious observation on the goal state with mean $\boldsymbol{\mu}_N$ and covariance $\boldsymbol{\mathcal{K}}_N$. This can be accomplished while still preserving the sparsity of the precision matrix [46]

$$\begin{aligned}\boldsymbol{\mathcal{K}}^{-1} &= \begin{bmatrix} & \widetilde{\boldsymbol{F}}^{-1} & \\ \boldsymbol{0} & \cdots \; \boldsymbol{0} & \boldsymbol{\mathrm{I}} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \widetilde{\boldsymbol{Q}}^{-1} & \\ & \boldsymbol{\mathcal{K}}_N^{-1} \end{bmatrix} \begin{bmatrix} & \widetilde{\boldsymbol{F}}^{-1} & \\ \boldsymbol{0} & \cdots \; \boldsymbol{0} & \boldsymbol{\mathrm{I}} \end{bmatrix} \\ &= \boldsymbol{G}^T\boldsymbol{Q}^{-1}\boldsymbol{G}. \end{aligned} \tag{3.13}$$

Since we do not want trajectories to deviate from the goal state, the covariance of the goal state $\boldsymbol{\mathcal{K}}_N$ should be relatively small thus keeping the desired goal state fixed. Figure 3.1 illustrates an example of the described GP prior suitable for motion planning in 2D spaces.

3.1.2 *GP interpolation*

A paramount advantage of representing continuous-time trajectories in motion planning with GPs is the opportunity to query the planned trajectory state $\boldsymbol{\theta}(\tau)$ at any time of interest τ, and not only at the discrete *support state* time instants. The exactly sparse block tridiagonal precision matrix supports computationally efficient, structure-exploiting GP interpolation

with $\mathcal{O}(1)$ complexity. State $\boldsymbol{\theta}(\tau)$ at $\tau \in [t_i, t_{i+1}]$ can be written as a function of only of its neighboring states [59]

$$\boldsymbol{\theta}(\tau) = \boldsymbol{\mu}(\tau) + \boldsymbol{\Lambda}(\tau)(\boldsymbol{\theta}_i - \boldsymbol{\mu}_i) + \boldsymbol{\Psi}(\tau)(\boldsymbol{\theta}_{i+1} - \boldsymbol{\mu}_{i+1}), \tag{3.14}$$

$$\boldsymbol{\Lambda}(\tau) = \boldsymbol{\Phi}(\tau, t_i) - \boldsymbol{\Psi}(\tau)\boldsymbol{\Phi}(t_{i+1}, t_i), \tag{3.15}$$

$$\boldsymbol{\Psi}(\tau) = \boldsymbol{Q}_{i,\tau}\boldsymbol{\Phi}(t_{i+1}, \tau)^T \boldsymbol{Q}_{i,i+1}^{-1}, \tag{3.16}$$

where $\boldsymbol{Q}_{a,b}$ is given in (3.12). Efficient GP interpolation enables keeping a relatively small number of *support states* even in cluttered environments with small obstacles, which reduces the computational burden compared to the discrete-time representation. It can also be exploited for providing a dense robot trajectory on the output, which can be directly executed without the need for post-processing.

3.1.3 *Constant-velocity motion model*

Finally, we represent robot dynamics with a constant-velocity motion model, which is a double integrator linear system with white noise injected in acceleration. The trajectory generated by (3.5) is defined by the Markovian state $\boldsymbol{\theta}(t)$ consisting of position and velocity in the robot's configuration space

$$\boldsymbol{\theta}(t) = \begin{bmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{bmatrix} \tag{3.17}$$

with the following system matrices

$$\boldsymbol{F}(t) = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \; \boldsymbol{L}(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}. \tag{3.18}$$

For the constant-velocity motion model the state transition matrix $\boldsymbol{\Phi}(t, s)$ evaluates to [81]

$$\boldsymbol{\Phi}(t, s) = \exp\{(t - s)\boldsymbol{F}(t)\} = \begin{bmatrix} \mathbf{1} & (t-s)\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \tag{3.19}$$

while the matrix $\boldsymbol{Q}_{a,b}$ defined in (3.12) evaluates to

$$\boldsymbol{Q}_{a,b} = \begin{bmatrix} \frac{\Delta t^3}{3}\boldsymbol{Q}_c & \frac{\Delta t^2}{2}\boldsymbol{Q}_c \\ \frac{\Delta t^2}{2}\boldsymbol{Q}_c & \Delta t \boldsymbol{Q}_c \end{bmatrix}. \tag{3.20}$$

This formulation provides a constant velocity GP centered around a zero acceleration trajectory. By applying this motion model, the joint accelerations are minimized, minimizing energy consumption in the process and fostering a physical meaning of trajectory smoothness. As an alternative, the constant acceleration motion model can be used [58] in specific cases where acceleration control is required. However, this comes with the expense of significantly increased computational cost since acceleration is added to the state vector in (3.17) and dimensions of all thereafter computed vectors and matrices increases. The constant velocity motion model is well-suited for motion planning in all experiments and applications we consider and we therefore rely on it throughout this thesis.

3.2 COLLISION AVOIDANCE

The ability to determine if a robot is in collision with objects in its environment is of paramount importance no matter the motion planning approach we rely on. An adequate robot model and environment representation are therefore a quintessential prerequisite for safe and efficient robot motion planning. In this thesis, we represent a robot as a set of spheres, while the Euclidean signed distance fields (SDFs) are used as environment representations. In the following subsections we describe the aforementioned representations and elucidate the reasons for choosing these in particular for collision checking within the proposed trajectory optimization methods.

3.2.1 *Robot model*

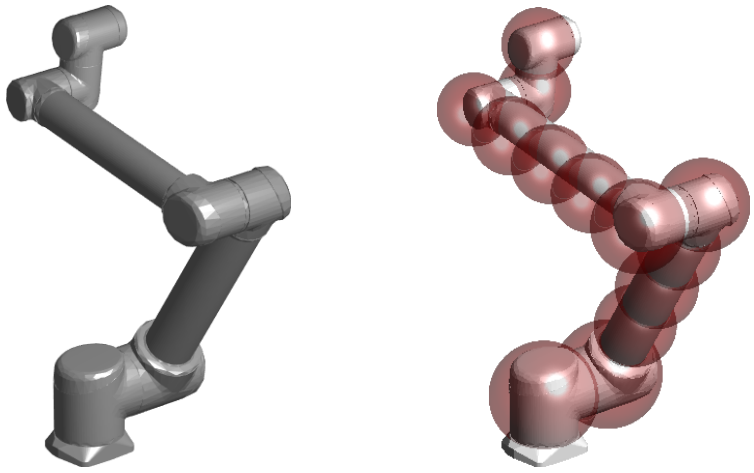
The robot configuration at a given time instant t_i is defined as a set of joint positions $\mathbf{q}_i$ in the configuration space $\mathbf{q}_i \in \mathcal{Q} \subset \mathbb{R}^n$ which comprises all feasible joint positions, where n is the number of DoF. Similarly, the robot's end-effector poses $\mathbf{x}$ form the task space $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^p$. The robot's forward kinematics function is defined as the following nonlinear mapping

$$\mathbf{f} : \mathcal{Q} \to \mathcal{X}. \tag{3.21}$$

The robot's body can be modelled as a set of arbitrary number of points, $\mathcal{B} \in \mathbb{R}^w, w \in \mathbb{N}$. Given a robot's configuration $\mathbf{q}_i$, the forward kinematics function can also provide the pose of a point $b \in \mathcal{B}$ of the robot body in Euclidean space.

In practical applications, the set of points $\mathcal{B}$ can be approximated with a set of geometric primitives (e.g., spheres, polyhedra, ellipsoids) that encompass the robot's body. Throughout this thesis, we deem the robot body as a set of spheres, which is a representation ubiquitously utilized in trajectory optimization methods [47, 71, 59]. In the case of a robotic arm, we fit a swept-sphere capsule to each robot arm link and then generate a discrete set of spheres for each capsule. This is illustrated in Fig. 3.2 where for the Universal Robots UR10 robot arm we show the mesh model that closely resembles the actual robot both with and without the corresponding set of spheres overlayed. For representing a mobile manipulator, we additionally model the base by fitting a minimal number of spheres with a diameter equal to the base height that enclose the base body. By using spheres as geometric primitives for robot modelling, the nearest distance from an obstacle to any point in the set $\mathcal{B}$ is simple to compute. The pertaining distance is given by the distance between an obstacle and the center of the sphere subtracted by the sphere radius. A trajectory $\boldsymbol{\theta}$ is then considered collision-free if for every configuration $\mathbf{q}_i \in \boldsymbol{\theta}$ the distance from each point b to any obstacle is greater than zero. Furthermore, checking if the robot is in a self-collision, i.e., if any of its links are in collision with any other link of the robot's body, simplifies to calculating the distance between two sphere centers and subtracting their radii.

While the set-of-spheres robot representation is not the most accurate way of modelling the robot's body, its simplicity enables very efficient collision checking that is crucial for achieving real-time operation with state-of-the-art algorithms on current hardware. Additionally, the mentioned simplicity implies that a set-of-spheres model can easily be manually constructed for different robots by a human operator, enabling fast deployment in practical applications. This particular robot representation makes the trajectory optimization



(a) Realistic mesh model

(b) A set-of-spheres representation

Figure 3.2: An example of representing the Universal Robots UR10 robot arm as a set of spheres. The robot can be adequately represented by only 13 spheres, which allows particularly efficient collision checking.

problems that we tackle tractable in practice and thus we relied on it for every experiment conducted in this thesis. However, note that neither of the herein proposed methods is restricted to using only a set-of-spheres representation. When accurate geometry modeling is required, robot representations based on meshes and different convex primitives might be better suited, although at higher computation costs. In such cases, any other collision checking tool, e.g., the Flexible Collision Library (FCL) [86], can be coupled with each of the proposed methods.

3.2.2 *Environment representation*

To enable real-time collision checking and consequently real-time motion planning, we need a way to efficiently query a data structure that contains information about the environment and locations of obstacles we want to avoid. Since we want to achieve high-dimensional motion planning, the chosen representation has to support accurate modelling of 3D environments while taking into account limited computational resources. The voxel occupancy grids may be the simplest and most intuitive way of representing a 3D environment. The 3D environment is discretized into a grid of voxels, which are units of information that define a point in the 3D space. Each voxel in the grid may contain a binary value that denotes occupancy, e.g., zeros indicate free space and ones indicate obstacles. Alternatively, the voxel occupancy grid may be probabilistic with each voxel encoding the probability of the pertaining point in a 3D environment being collision-free. Occupancy grids have been omnipresent in methods for 2D navigation and SLAM problems [87, 88, 89, 90], where pixels are used instead of voxels. The downside of using voxel occupancy grids is that they do not provide any information on distance between the robot and obstacles and thus do not provide first-order information or allow for predictive behavior. If the robot is very close to an obstacle, the information provided by voxel occupancy grids is the same as

if the robot was very distant from every obstacle. Including the information on distance from nearest obstacles in trajectory optimization can enable safer collision avoidance and increase human comfort in human-robot interaction.

While we can compute the distance from obstacles using the Euclidean distance transform [91], its calculation may become too computationally demanding for fast motion planning, especially for large environments with many obstacles. For this reason, in Chapters 4 and 5 we will opt for precomputing the signed distance for each voxel in the provided occupancy grid of desired resolution and constructing the SDF. Using signed distance not only provides information about the distance from the closest obstacle if the robot is collision-free, but also provides a measure of how much the robot body is penetrating the obstacle if they are colliding. The precomputed SDF can be queried in real-time to obtain information about the signed distance at a given robot's position, which is then utilized to construct the obstacle cost used by the trajectory optimization methods proposed within this thesis. SDFs are independent of the environment complexity and each check for collision requires constant time. They are thus suitable for collision checking for robots of complex kinematic structure and with high DoF. Additionally, while SDFs are a discrete representation, we can easily make a query for any position in robot's task space by using trilinear interpolation on the voxel grid.

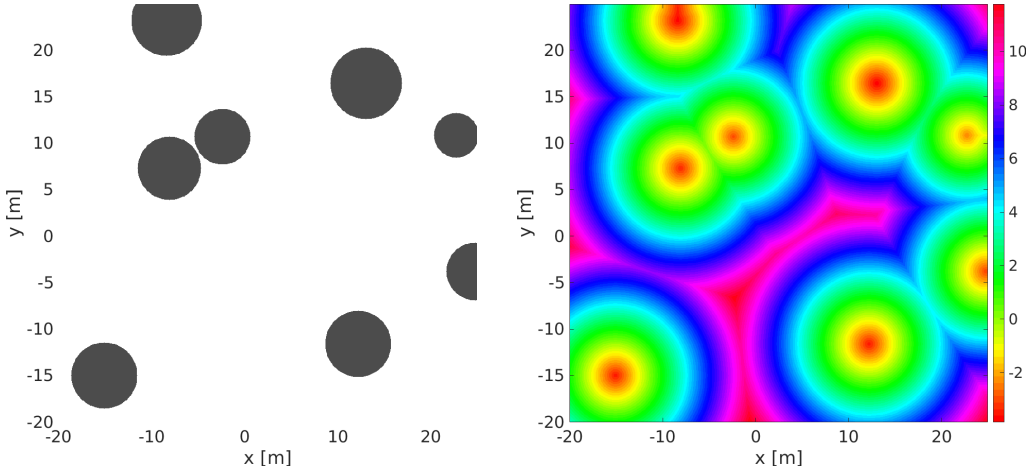
The precomputed SDFs provide a fast and accurate distance information for static environments. However, in Chapter 6 we will consider robot motion planning in dynamic environments with human presence. In such cases, computing an SDF from scratch each time an update occurs in the voxel occupancy grid is too computationally demanding and makes real-time operation infeasible. Recently, several methods that tackle this issue have been proposed, either relying on a GPU to construct distance fields in real time directly from live sensor data [92, 93, 94] or incrementally updating the SDF after changes in the environment map have occurred [95, 96, 97]. In Chapter 6 we will rely on one such method, namely GPU-Voxels [94], in order to obtain a real-time SDF representation from RGB-D live sensor data as a principal component in the proposed motion planning framework that supports robot operation in real-world dynamic environments.

3.2.3 *Collision checking*

After obtaining adequate representations for the robot model and the environment, we need to utilize them in a collision checking procedure that will foster efficient trajectory optimization and lead to safe motion plans. We construct the obstacle cost function that is deployed as a component of the optimization criteria throughout this thesis. This obstacle cost function for any robot configuration $\boldsymbol{\theta}_i$ is evaluated by computing the hinge loss for each sphere of the robot model and collecting it into a single scalar

$$f[\boldsymbol{\theta}_i] = \sum_{j=1}^{S} c\Big[d\big(\boldsymbol{x}(\boldsymbol{\theta}_i, S_j), \mathcal{O}\big)\Big], \tag{3.22}$$

where $\boldsymbol{x}$ is the forward kinematics of a robot, $\mathcal{O}$ is the set of obstacles in the environment, d is the Euclidean signed distance function, c is the hinge loss function and S is the number of spheres that accurately represent the robot model. Given a sphere radius and its center



(a) A simple 2D environment (b) The corresponding SDF

Figure 3.3: An example of a 2D random forest synthetic environment and the corresponding SDF. Querying the SDF at the robot's position in the environment returns the signed distance from the closest obstacle. If this distance is negative, the robot would be in collision with an obstacle at that position.

position, we query the SDF, providing a precomputed Euclidean signed distance from the sphere at position $\boldsymbol{x}$ to the closest obstacle surface in the task space. The aforementioned hinge loss function is defined as

$$\boldsymbol{c}(d) = \begin{cases} \varepsilon - d & \text{if } d \leq \varepsilon \\ 0 & \text{if } d > \varepsilon \end{cases}, \tag{3.23}$$

where d is the calculated signed distance between the sphere and the closest obstacle, and ε is the parameter that can be seen as a safety distance indicating the boundary of the area near obstacle surfaces that is considered dangerous [46]. Even if a given trajectory is collision-free but passes too close to the obstacles, its incurred obstacle cost would be non-zero. The optimization thus guides the robot to stay a minimum distance ε away from obstacles.

3.3 Kinematic singularity avoidance

When performing a desired task, robotic arms often have additional constraints induced by the task (e.g., obstacle avoidance) which may restrict their movement in task space. Such constraints may incur non-optimal mapping of some configurations from the configuration space to the task space, resulting in singular configurations, i.e., *singularities*. Singularities hinder task space mobility, requiring particularly large joint motions which can result in perilous joint movement or task execution failure. It is crucial to preserve the ability to move in every direction in task space in order to ensure robot's agility in case of encountering unforeseeable changes during task execution. Therefore, recognizing and avoiding singular configurations during motion planning has been a topic of robotics research for decades [98, 99, 100, 101].

Singularities can be identified by observing singular values of the robot's Jacobian matrix, which is a function that maps configuration space velocities to task space velocities [102]. There exist several indices that try to capture the kinematic sensitivity by relying on singular values of the Jacobian. The manipulability index [98] measures the volume of the manipulability ellipsoid and is perhaps the most widely employed way of recognizing singular configurations. Another commonly used measure is the dexterity index [103] that calculates the ratio between the longest and shortest manipulability ellipsoid axes. These indices have been successfully integrated with inverse kinematics [100, 101] and motion planning [104, 105, 69, 106] by looking to maximize the volume of the manipulability ellipsoid in order to avoid singular configurations. Nevertheless, both the manipulability and dexterity index possess shortcomings due to their inability to fully capture the geometry of the manipulability ellipsoid. Redundancy resolution approaches such as [107] do not explicitly consider manipulability or dexterity indices and have long been used for singularity avoidance. However, they can be prone to algorithmic singularities caused by conflicting objectives and are therefore employed only for simple task hierarchies. Rozo et al. used Stein divergence to derive a geometry-aware similarity measure between two manipulability ellipsoids [108] and employed it in the context of task learning. To this end we present the *geometry-aware singularity index* [109], which relies on a Riemannian metric on the manifold of symmetric positive definite matrices to measure the proximity of a configuration to singularity, bypassing the failure modes inherent to manipulability and dexterity. In the following subsections we describe kinematic singularities and the proposed novel geometry-aware singularity index that we rely on in Chapter 5 for identifying and avoiding kinematic singularities within our trajectory optimization method to achieve safe motion plans, retaining the mobility of robot arms in all task-space directions throughout the movement.

3.3.1 *Kinematic singularities*

Following the forward kinematics function defined in (3.21), the robot's inverse kinematics function is conversely defined as a mapping from the robot's task space to its configuration space

$$\mathbf{f}^{-1} : \mathcal{X} \rightarrow \mathcal{Q}. \tag{3.24}$$

The associated Jacobian matrix is obtained by finding the gradient of $\mathbf{f}$ with respect to joint values $\mathbf{q}_i$,

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}_i}. \tag{3.25}$$

The Jacobian matrix defines kinematic relationship between configuration and task space velocities [102]

$$\dot{\mathbf{x}}_i = \mathbf{J}\dot{\mathbf{q}}_i, \tag{3.26}$$

where $\mathbf{J}$ is the Jacobian matrix computed at configuration $\mathbf{q}_i$, while $\dot{\mathbf{q}}_i$ and $\dot{\mathbf{x}}_i$ are the joint and task space velocities.

The mapping of a unit sphere in the space of joint velocities $\|\dot{\mathbf{q}}_i\|^2 = 1$ to the task velocity space can be defined as

$$\|\dot{\mathbf{q}}_i\|^2 = \dot{\mathbf{x}}^T \left(\mathbf{J}\mathbf{J}^T\right)^{-1} \dot{\mathbf{x}}. \tag{3.27}$$

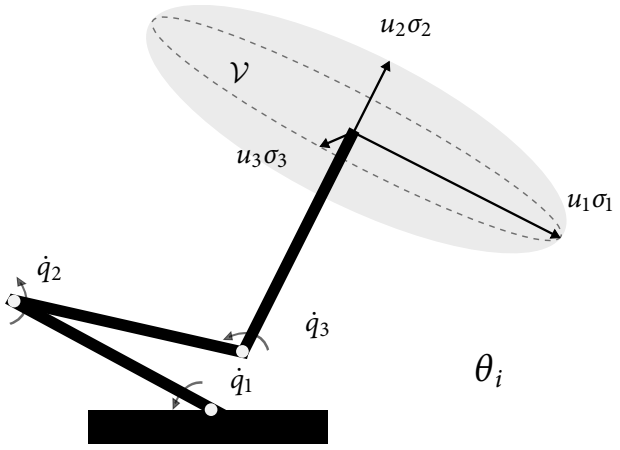


Figure 3.4: Illustration of the manipulability ellipsoid of volume $\mathcal{V}$ for a manipulator end-effector at configuration $\boldsymbol{\theta}_i$. Larger axis lengths indicates higher mobility.

The conditioning of the matrix $\mathbf{J}\mathbf{J}^T$ governs the scaling of joint velocities to the task space. Therefore, the eigenvalues of this matrix provide information about directional mobility of the robot arm in some configuration $\mathbf{q}_i$. The matrix $\mathbf{J}\mathbf{J}^T$ in Eq. (3.27) is symmetric positive semi-definite (SPD) and is termed the manipulability ellipsoid of the end-effector [98]

$$\mathbf{M}(\mathbf{q}_i) = \mathbf{J}\mathbf{J}^T. \tag{3.28}$$

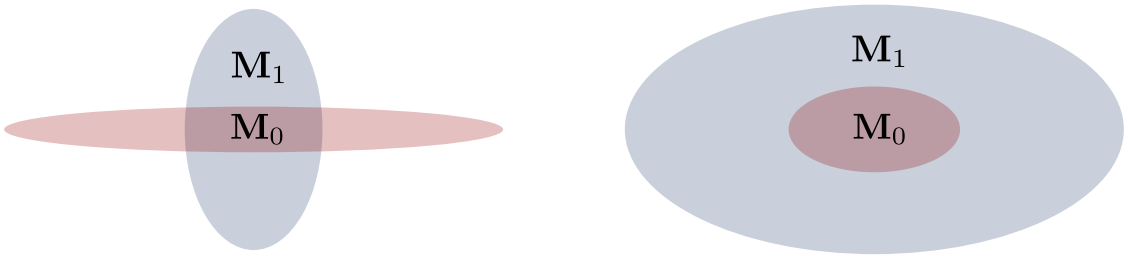
By computing the singular value decomposition (SVD) of the Jacobian, $\mathbf{J} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, we can obtain the principal axes $\sigma_0\mathbf{u}_0, \sigma_1\mathbf{u}_1, \ldots, \sigma_p\mathbf{u}_p$. The length and orientation of these axes determine directions in which higher values of task space velocities can be achieved. This is illustrated in Figure 3.4, depicting that larger principal axes lengths lead to higher robot mobility. The eigenvalues of the manipulability ellipsoid $\mathbf{M}$ are equal to the squared singular values σ^2 of the Jacobian matrix $\mathbf{J}$. This implies that when one or more eigenvalues of $\mathbf{M}$ become zero, the Jacobian matrix is poorly conditioned and thus non-invertible, i.e. the robot is in a singular configuration.

3.3.2 *Manipulability index*

As a consequence of the manipulability ellipsoid eigenvalues corresponding to the squared singular values of the Jacobian, all configurations for which the Jacobian matrix is non-invertible have a corresponding manipulability ellipsoid with at least one degenerate axis. Movements of the robot end-effector that begin from these singular configurations are susceptible to high joint velocities and unwanted dynamic characteristics. To detect singular configurations, we can observe singular values of the Jacobian matrix, which is a basis of several developed indices that attempt to provide a measure of proximity of a configuration to singularity. One of the indices commonly used for singularity avoidance and improving kinematic sensitivity is the so-called manipulability index, defined as

$$m = \sqrt{\det(\mathbf{J}\mathbf{J}^T)}. \tag{3.29}$$

The value of m is in proportion to the volume of the manipulability ellipsoid defined in Eq. (3.28), providing an intuitive geometric interpretation of the manipulability index. Singu-



(a) (b)

Figure 3.5: An example of failure modes for common singularity indicators. a) The manipulability index fails in configurations featuring elongated ellipsoids with near zero-length axes. b) The dexterity index fails when the ellipsoid is of an uniformly small scale.

larities imply that the corresponding manipulability ellipsoids have at least one degenerate axis, resulting in zero-volume manipulability ellipsoids. Consequently, the manipulability index of singularities is zero and can thus be used for singularity detection in inverse kinematics [101] and trajectory optimization [69]. However, there exist cases where manipulability index is ineffective in detecting configurations that are close to being singular. This stems from the fact that the volume of the manipulability ellipsoid can remain relatively large near singularities if other axes of the manipulability ellipsoid are particularly elongated. An example of such case is shown in Fig. 3.5a, where two ellipsoids have the same volume, despite one ellipsoid possessing one axis of a near-zero length and the corresponding configuration being close to singular.

3.3.3 *Dexterity index*

Another established index commonly used for singularity detection and avoidance is the dexterity index [103], defined as

$$\kappa = \frac{\sigma_{max}}{\sigma_{min}}, \tag{3.30}$$

where σ_{max} and σ_{min} denote the maximal and minimal singular values of the Jacobian matrix. It follows that the dexterity index is essentially a measure of difference in relative lengths between the longest and shortest axis of the manipulability ellipsoid, where large values indicate an elongated shape. When the length of one axis of the manipulability ellipsoid is near-zero, the dexterity index may be high and can thus in most cases be used for singularity detection. However, the drawback of the dexterity index is its incapability to inform about the scale of the ellipsoid. This problem manifests when a configuration is far from singularity but has one axis of a particularly large length – the dexterity index cannot distinguish such configuration from a singular one. An example of such case is shown in Fig. 3.5b, where two ellipsoids have the same difference in relative lengths between the two axes, despite one ellipsoid having smaller volume and being closer to singularity.

3.3.4 *A geometry-aware singularity index*

The manipulability and dexterity indices are commonly employed for singularity detection and avoidance. However, as described in prior sections, both of them have drawbacks that

make them inapplicable in certain cases. In this thesis, we will therefore utilize a recently developed geometry-aware singularity index [109] that is defined as

$$\xi \coloneqq \left\| \log\left(\boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{M} \boldsymbol{\Sigma}^{-\frac{1}{2}} \right) \right\|_F^2 , \tag{3.31}$$

where $\mathbf{M}$ is the manipulability ellipsoid of a given configuration, $\boldsymbol{\Sigma}$ is an arbitrary reference ellipsoid and $\|\cdot\|_F$ is the Frobenius norm. The index defined in Eq. (3.31) is essentially a squared Riemannian distance between a reference ellipsoid $\boldsymbol{\Sigma}$ and manipulability ellipsoid M. For a more in-depth treatment of Riemannian geometry in the context of manipulability ellipsoids and singularity avoidance the reader is referred to [110], [109]. By carefully choosing the reference ellipsoid $\boldsymbol{\Sigma}$ in Eq. (3.31), the singularity index gains the notion of measuring proximity of a given configuration to a singular one.

We select the reference ellipsoid $\boldsymbol{\Sigma}$ to be a sphere with the radius greater than the length of the longest manipulability ellipsoid axis, thus enclosing the manipulability ellipsoid

$$\boldsymbol{\Sigma} = k\boldsymbol{I},\ k \geq \sigma_{\max}^2 . \tag{3.32}$$

This choice of k results in ξ decreasing with the increase of the singular values of the Jacobian, since the manipulability ellipsoid is always inside the reference ellipsoid $\boldsymbol{\Sigma}$. Due to the symmetry of the sphere, the geometry-aware singularity index is invariant to different orientations of the manipulability ellipsoid [109]. This trait is advantageous in singularity avoidance since the orientation of the manipulability ellipsoid does result in changes to singular values of the Jacobian. When a robot arm changes its configuration, the corresponding manipulability ellipsoid changes its size, shape and orientation. In contrast to manipulability and dexterity indices, the geometry-aware singularity index encodes information about ellipsoid's axis lengths, shape and orientation relative to some reference ellipsoid. This property allows us to avoid issues that arise when applying the conventional indices in planning the robot's motion, which encode only partial information about the manipulability ellipsoid structure. In Chapter 5 we will exploit the geometry-aware singularity index for devising an optimization cost that can be included in the presented trajectory optimization in order to generate robot trajectories that retain the mobility in all task-space directions throughout the movement.

3.4 SUMMARY

In this chapter, we presented the theoretical background and mathematical tools that form the foundation of high-dimensional motion planning methods based on trajectory optimization that we introduce in the following sections. We introduced continuous-time GPs as trajectory representations and the special GP prior suitable for motion planning with the exactly-sparse precision matrix structure. The mentioned GP prior allows for querying the robot trajectory at any time of interest due to the sparsity-exploiting efficient interpolation, which can be utilized for dense collision checking or in order to directly execute the obtained trajectory without the need for post-processing. Moreover, we presented the set-of-spheres as a robot model that is simple to construct and allows for particularly efficient checking of both self-collisions and collisions between the robot's body and the environment it operates in. We also introduced SDFs as environment representations that are independent of environment complexity and support constant-time queries, while providing the distance information. In conjunction, the set-of-spheres model and SDFs enabled us to devise an obstacle cost function that will be used within proposed trajectory optimization methods for collision avoidance. In the end, kinematic singularities were discussed as a potential hindrance to ensuring robot's agility in case of encountering unforeseeable changes during task execution. A novel geometry-aware singularity index was presented that is able to successfully identify kinematic singularities even in specific cases in which the commonly-used manipulability and dexterity indices fail.

4

Motion planning in cluttered environments

This chapter presents a novel approach to high-dimensional robot motion planning in cluttered environments. The main idea behind the presented approach is to use a heteroscedastic GP prior that is specifically targeted for improved performance in collision avoidance in complex environments coupled with a cross-entropy based stochastic trajectory optimization method that is often able to escape the infeasible local minima which are omnipresent in such environments. We evaluate the proposed approach in three simulated scenarios: a maze benchmark, a 7 DoF robot arm planning benchmark and a 10 DoF mobile manipulator trajectory planning example and compare it to two state-of-the-art trajectory optimization methods, namely the GPMP2 [71] and STOMP [71] algorithms. Our results demonstrate the following: (i) the proposed method yields a more thorough exploration of the solution space in complex environments than the current state of the art, while having comparable execution time, (ii) the introduced heteroscedasticity generates GP priors better suited for collision avoidance and (iii) the proposed method has the ability to efficiently tackle high-dimensional trajectory planning problems.

4.1 Introduction

Although state-of-the-art trajectory optimization methods generate fast solutions in high-dimensional spaces, they have limited exploration ability, and in complex environments often converge to infeasible local minima, since in most cases they rely on gradient-based optimization techniques. In general optimization problems, gradient-free stochastic optimization is often employed for problems with plethora of local minima and is therefore the approach that we will rely on for the method proposed in this chapter with the aim of improving trajectory optimization performance in cluttered environments that are challenging for traditional gradient-based methods. In motion planning, this is a relatively unexplored direction with only a few existing works. A recently proposed particle swarm filter for trajectory optimization [77] showed better local minima avoidance than gradient-based methods; however, it did not demonstrate effectiveness in complex, highly cluttered environments and carries the aforementioned problems associated with discrete-time trajectory representations. In [79, 80], the authors proposed a general randomized path planning method based on sampling in the space of trajectories. At its core lies the cross-entropy method for estimation of rare-event probabilities. However, this approach relies on sampling from a probability distribution over the set of feasible paths – sampling whole trajectories that are

collision-free, which can become infeasible in cluttered environments or high-dimensional spaces.

In this chapter, we propose a gradient-free stochastic optimization method for trajectory planning with continuous-time GP trajectory representations. The proposed method is supplemented by our two research papers [111, 112]. The contributions of the herein presented method stem from introducing the heteroscedastic GP, together with exploited sparsity for efficient covariance estimation, and a cross-entropy based stochastic optimization for importance sampling based trajectory optimization for motion planning. The main advantages of the proposed method are

(i) continuous time representation leading to smooth trajectories that can be queried at any time of interest,

(ii) powerful heteroscedastic priors better suited for collision avoidance in motion planning problems,

(iii) estimation of the collision-free trajectories distribution for a given environment allowing for faster convergence,

(iv) stochastic trajectory optimization allowing for better exploration while being less prone to local minima than gradient-based methods.

While our method belongs to the family of trajectory optimization approaches, it relies on random trajectory samples which raises a connection to the sampling based planning, with the difference that we sample whole trajectories, while sampling-based methods sample robot states. The proposed method is an example of bridging the gap between sampling based and trajectory optimization approaches in order to generate fast solutions in high dimensional spaces while retaining the ability to thoroughly explore the environment. We evaluated our method on three simulation scenarios: a maze benchmark, a 7 DoF robot arm planning benchmark and a 10 DoF mobile manipulator trajectory planning example. In the benchmark examples we compared the proposed method to GPMP2 and STOMP, which are state-of-the-art gradient-based trajectory optimization methods, while the mobile manipulator example showcases the applicability of the proposed method in high-dimensional planning problems. The results show that the proposed method yields a higher success rate in complex environments with comparable execution time.

4.2 Heteroscedastic GPs as trajectory representations

A continuous-time GP robot trajectory representation allows efficient reasoning about collisions and querying the robot state at any time of interest. For the method presented in this chapter, we rely on the GP trajectory representation framework for motion planning presented in Section 3.1 which includes constructing a GP prior suited for motion planning and an efficient GP interpolation procedure. In this section, we expand on the mentioned theoretical background by introducing several mathematical tools relevant for the herein proposed trajectory optimization method. We introduce heteroscedasticity in the time-domain of the proposed GP and discuss its benefits with regards to motion planning

problems. We also provide a principled way to draw trajectory samples from the pertaining GP and a way to efficiently construct the full covariance matrix of the employed GP.

4.2.1 *Heteroscedastic GP prior for motion planning*

As elaborated in Section 3.1, the GP prior suited for motion planning is generated by an LTV-SDE in (3.5) with the underlying white noise process in (3.6). However, the crucial difference in approach discussed in this chapter is that the covariance $\boldsymbol{Q}_c(t)$ is time-varying and consequently generates a heteroscedastic GP [68]. In other words, the covariance $\boldsymbol{Q}_c(t)$ of the white noise process $\boldsymbol{w}(t)$ is non-constant through time

$$\boldsymbol{w}(t) \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{Q}_c(t)\delta(t-t')), \tag{4.1}$$

where $\boldsymbol{Q}_c(t)$ is a positive semi-definite time-varying power-spectral density matrix. The novel heteroscedastic GP trajectory representation introduces several benefits for motion planning, which we discuss in Section 4.2.2.

While the mean vector $\boldsymbol{\mu}$ and the precision matrix $\boldsymbol{\mathcal{K}}^{-1}$ fully determine a continuous-time GP in (3.1), in some situations the covariance matrix $\boldsymbol{\mathcal{K}}$ might still be needed, i.e. for data visualization or when optimizing trajectories while learning from demonstration [68]. Since the method we propose in this chapter employs GPs to estimate trajectory distributions, for completeness we present a sparsity-exploiting computation method of the covariance matrix from the precision matrix. A naive approach is to compute $\boldsymbol{\mathcal{K}}$ by inverting the precision matrix. However, due to the exact sparsity of the precision matrix $\boldsymbol{\mathcal{K}}^{-1}$, the covariance matrix $\boldsymbol{\mathcal{K}}$ can be computed efficiently [113]. First we carry out a sparse lower-diagonal-upper decomposition,

$$\boldsymbol{\mathcal{K}}^{-1} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T, \tag{4.2}$$

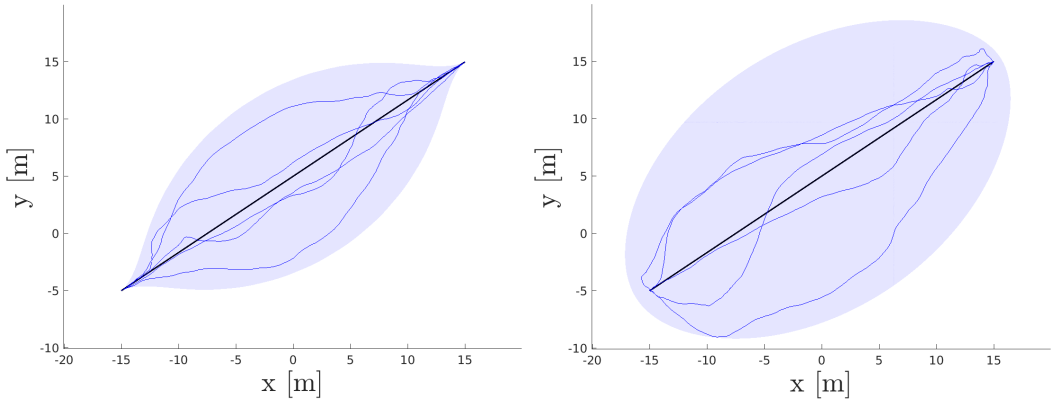
where $\boldsymbol{D}$ is diagonal and $\boldsymbol{L}$ is lower-triangular with ones on the main diagonal. Since $\boldsymbol{\mathcal{K}}$ is symmetric, only the calculation of the main diagonal and the lower-half blocks is required. Blocks of the covariance matrix indexed with j, k can then be computed through backward substitution

$$\boldsymbol{\mathcal{K}}_{N,N} = \boldsymbol{D}_{N,N}^{-1}, \tag{4.3}$$
$$\boldsymbol{\mathcal{K}}_{N,N-1} = -\boldsymbol{\mathcal{K}}_{N,N}\boldsymbol{L}_{N,N-1}, \tag{4.4}$$
$$\boldsymbol{\mathcal{K}}_{N-1,N-1} = \boldsymbol{D}_{N-1,N-1}^{-1} - \boldsymbol{\mathcal{K}}_{N-1,N}\boldsymbol{L}_{N,N-1}, \tag{4.5}$$
$$\vdots \tag{4.6}$$
$$\boldsymbol{\mathcal{K}}_{j,k} = \delta(j,k)\boldsymbol{D}_{j,k}^{-1} - \sum_{l=k+1}^{N} \boldsymbol{\mathcal{K}}_{j,l}\boldsymbol{L}_{l,k} \quad (j \geq k). \tag{4.7}$$

For a more in-depth treatment of this covariance matrix calculation procedure, the reader is referred to [113, 114].

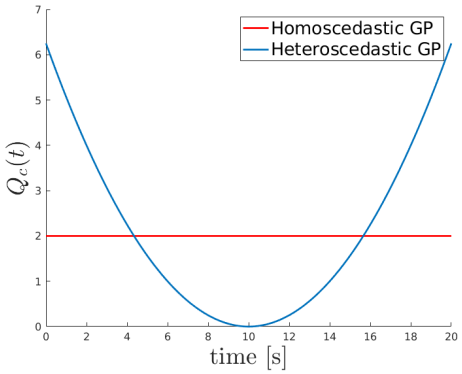
Since the proposed trajectory optimization method is stochastic in its nature, it relies on sampling trajectories from the presented heteroscedastic GP trajectory distribution. A sample trajectory can be generated using [115]

$$\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{A}\boldsymbol{Z}, \tag{4.8}$$



(a) Homoscedastic GP

(b) Heteroscedastic GP



(c) White noise process $\boldsymbol{w}(t)$ covariance through time

Figure 4.1: Comparison of the homoscedastic and the proposed heteroscedastic GP priors which depend on the covariance of the white noise process $\boldsymbol{w}(t)$

where $\boldsymbol{A}$ is a lower triangular matrix obtained by Cholesky decomposition of the covariance matrix, $\boldsymbol{\mathcal{K}} = \boldsymbol{A}\boldsymbol{A}^T$, and $\boldsymbol{Z}$ is a vector of N standard normal variables $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The sampling process in (4.8) requires knowledge of the covariance matrix $\boldsymbol{\mathcal{K}}$. While calculation of the covariance matrix $\boldsymbol{\mathcal{K}}$ employing the procedure described in (4.3)-(4.7) is efficient in comparison to the straightforward inversion of the precision matrix, it can still be computationally expensive in higher-dimensional state-spaces with dense parameterization of the GP. Alternatively, a sample trajectory can be generated without computing the covariance matrix [115]

$$\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{B}^{-T}\boldsymbol{Z}, \tag{4.9}$$

where $\boldsymbol{B}$ is a lower triangular matrix obtained by Cholesky decomposition of the precision matrix, $\boldsymbol{\mathcal{K}}^{-1} = \boldsymbol{B}\boldsymbol{B}^T$, and $\boldsymbol{Z}$ is a vector of N standard normal variables $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Since the precision matrix $\boldsymbol{\mathcal{K}}^{-1}$ has block-tridiagonal structure, its Cholesky decomposition leads to a lower bidiagonal matrix which can be efficiently inversed with exploited sparsity.

4.2.2 *Benefits of proposed heteroscedasticity*

State-of-the-art GP-based trajectory optimization methods [58, 59, 46] work by minimizing the sum of two costs: the obstacle cost and the smoothness cost. The obstacle cost measures how close the trajectory is to obstacles, while the smoothness cost measures the deviation of the trajectory from the GP prior mean. Those methods use covariance as a parameter in optimization, with smaller values of $\mathcal{K}$ penalizing trajectory deviation from the prior mean. For therein employed homoscedastic GPs, small constant values of the white noise process covariance $\boldsymbol{Q}_c$ allow high variance of states near the midpoint of trajectory, while states close to the trajectory start or goal have relatively small variance. If the prior mean of those states passes through an obstacle, the Levenberg-Marquardt optimization, utilized in [59, 46], will have difficulty escaping the collision. The collision happens when the incurred smoothness cost becomes relatively large in comparison to obstacle cost thus creating a local minimum. Figure 4.1a shows that sample trajectories drawn from a homoscedastic GP do not have large deviation from the mean near the first and the last state. With larger constant values of $\boldsymbol{Q}_c$ this problem diminishes; however, states near the trajectory midpoint then have too high variance and trajectories lose the desirable smoothness property.

By introducing heteroscedasticity of the underlying white noise process, we can design GPs that are better suited for motion planning problems and help alleviate the problem of local minima. Ideally, the GP should be able to generate trajectories that maneuver around the obstacles near start and goal states, while retaining the smoothness property in the middle. With careful selection of the proposed time-varying white noise power-spectral density matrix $\boldsymbol{Q}_c(t)$, we are able to model GPs that achieve the stated goal. We propose modelling $\boldsymbol{Q}_c(t)$ as an isotropic matrix $\boldsymbol{Q}_c(t) = Q_c(t)\boldsymbol{I}$ with parameter $Q_c(t)$ modelled as a parabola. A parabola has high values at the beginning and end, while the lowest value lies at the temporal midpoint of the trajectory, leading to GPs that have the aforementioned desirable features. Note that the GP covariance $\mathcal{K}(t, t')$ is obtained by propagating $Q_c(t)$ with the underlying motion model, as defined in (3.9), and thus low (or even zero) value of $Q_c(t)$ at a particular time instant does not imply small GP covariance. An example of such $Q_c(t)$ is depicted in Figure 4.1c and it leads to a heteroscedastic GP shown in Figure 4.1b. Notably, sample trajectories drawn from the example heteroscedastic GP have large deviation from the mean near the first and the last state. The example GP would be able to thoroughly explore the environment and generate trajectories that bypass obstacles near the first and the last state. Note that any non-negative function can be used for $Q_c(t)$, depending on the specific context of the motion planning problem at hand. For example, one could model $Q_c(t)$ as a monotonically decreasing function, resembling the aim of exploring more at the beginning of the trajectory and less towards the end.

4.3 Cross-entropy based trajectory optimization

In this section we describe the proposed gradient-free stochastic optimization method for motion planning. First, we formalize the motion planning problem as trajectory optimization with trajectories considered as samples from the heteroscedastic GP proposed in Section 4.2. Then, we present the cross-entropy optimization method adapted to the motion planning problem setting and derive the GP mean and covariance update rules. Finally, we

state the proposed algorithm steps and give remarks regarding its computational efficiency.

4.3.1 *Problem formulation*

Formally, the goal of trajectory optimization for motion planning is to find a smooth, collision-free trajectory through the robot's configuration space between the start and end points. Prior work in this area [47, 45, 59] models the cost of a trajectory using two terms: a prior term, which usually encodes smoothness that minimizes higher-order derivatives of the robot states, and an arbitrary state-dependent term, which usually measures the cost of being near obstacles. However, our optimization criteria consists solely of an arbitrary state-dependent cost term. We reason that our trajectory carries an inherent property of smoothness since we model it as a sample from the GP defined in (3.1) with the underlying constant-velocity model described in Section 3.1.3. Therefore, our method starts with the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\theta}(t)}{\text{minimize}} \quad & f[\boldsymbol{\theta}(t)] \\ \text{subject to} \quad & \boldsymbol{\theta}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \mathcal{K}(t, t')). \end{aligned} \tag{4.10}$$

The state-dependent cost term $f[\boldsymbol{\theta}(t)]$ can include any cost function corresponding to the desired trajectory properties. For example, the state-dependent cost term can incorporate collision avoidance, trajectory smoothness, various task-space constraints (e.g. holding the cup of coffee upright) [59], motor torques [71] and manipulability [69]. In the scope of this chapter we consider only collision avoidance. The reasoning behind not taking into account other cost functionals is twofold. First, the heteroscedastic GP proposed in Section 4.2 with the corresponding power-spectral density matrix $\boldsymbol{Q}_c(t)$ purposefully increases covariance near the first and the last state. While significantly improving collision avoidance in these areas, this can hinder optimization of other criteria by imposing larger accelerations or motor torques near the first and the last state. Second, the cross-entropy based trajectory optimization method relies on sampling a batch of trajectories and then evaluating their cost. If there are multiple optimization criteria, the incurred computational cost can become too high and make the method sample inefficient. For encoding the collision avoidance criterion, we rely on the obstacle cost function defined in (3.22), with robot's body represented as a set of spheres as explained in Section 3.2.1 and an SDF environment representation presented in Section 3.2.2.

4.3.2 *Cross-entropy optimization method*

Most of the state-of-the-art approaches use gradient-based methods which find locally optimal trajectories. In this chapter, we instead optimize using a derivative-free cross-entropy stochastic optimization method. This allows for better exploration of the trajectory space while being less prone to local minima. Herein, we provide a brief introduction to the stochastic framework for solving the optimization problem in (4.10) by sampling trajectories from a GP in (3.1). Our development follows [116], but with a GP instead of a general parametric distribution and with notation adapted to our setting.

Consider the optimization problem in (4.10). Let us denote its minimum by γ^*. We can thus write

$$f[\boldsymbol{\theta}^*(t)] = \gamma^* = \min_{\boldsymbol{\theta}(t)\sim\mathcal{GP}(\boldsymbol{\mu},\mathcal{K})} f[\boldsymbol{\theta}(t)]. \tag{4.11}$$

The starting point of the cross-entropy method is associating an estimation problem with the optimization problem in (4.11). First, a collection of indicator functions $I_{\{f[\boldsymbol{\theta}]\leq\gamma\}}$ for various thresholds $\gamma \in \mathbb{R}$ is defined

$$I_{\{f[\boldsymbol{\theta}]\leq\gamma\}} = \begin{cases} 1 & \text{if } f[\boldsymbol{\theta}] \leq \gamma \\ 0 & \text{if } f[\boldsymbol{\theta}] > \gamma \end{cases}. \tag{4.12}$$

For a certain GP defined by its mean $\boldsymbol{\mu}$ and covariance $\mathcal{K}$ let us associate the optimization problem in (4.11) with the problem of estimating the probability that a cost function $f(\boldsymbol{\theta})$ has smaller value than a given threshold γ

$$\begin{aligned} \iota(\gamma) &\coloneqq \mathbb{P}_{\boldsymbol{\mu},\mathcal{K}}(f[\boldsymbol{\theta}] \leq \gamma) \\ &= \int_{\boldsymbol{\theta}} I_{\{f[\boldsymbol{\theta}]\leq\gamma\}} p(\boldsymbol{\theta};\boldsymbol{\mu},\mathcal{K}) d\boldsymbol{\theta} \\ &= \mathbb{E}_{\boldsymbol{\mu},\mathcal{K}}[I_{\{f[\boldsymbol{\theta}]\leq\gamma\}}], \end{aligned} \tag{4.13}$$

where $\mathbb{P}_{\boldsymbol{\mu},\mathcal{K}}$ denotes the probability measure under which the random trajectory $\boldsymbol{\theta}$ has probability density function $p(\boldsymbol{\theta};\boldsymbol{\mu},\mathcal{K})$ and $\mathbb{E}_{\boldsymbol{\mu},\mathcal{K}}$ is the corresponding expectation operator. To see how the problems in (4.13) and (4.11) are associated, suppose that a certain threshold γ is close to the optimal value γ^*. In a typical case the probability of a randomly sampled trajectory $\boldsymbol{\theta}$ being near the optimal one is rather small, thus $f[\boldsymbol{\theta}] \leq \gamma$ is a rare event. Note that, for a globally optimal $\boldsymbol{\theta}^*$, $\iota(\gamma^*) = p(\boldsymbol{\theta}^*;\boldsymbol{\mu},\mathcal{K})$. Therefore, for $\gamma = \gamma^*$ a natural way to estimate $\iota(\gamma)$ would be to use the rare-event estimation framework [117] which can solve this problem efficiently by making adaptive changes to the cross-entropy and thus iteratively generate GPs that are *steered* in the direction of the theoretically optimal density.

This is achieved by employing the likelihood-ratio estimator [116] with reference parameters $\{\boldsymbol{\mu},\mathcal{K}\}$ given by

$$\boldsymbol{\mu}^*,\mathcal{K}^* = \operatorname*{argmax}_{\boldsymbol{\mu},\mathcal{K}} \mathbb{E}_{\boldsymbol{\mu},\mathcal{K}}[I_{\{f[\boldsymbol{\theta}]\leq\gamma\}} \ln p(\boldsymbol{\theta};\boldsymbol{\mu},\mathcal{K})]. \tag{4.14}$$

In other words, we try to find the GP parameters $\boldsymbol{\mu}^*,\mathcal{K}^*$ that maximize the expectation of finding the optimal trajectory $\boldsymbol{\theta}^*$ which minimizes the cost function $f(\boldsymbol{\theta})$. In motion planning context, our approach can be seen as trying to find a GP that is the best approximation (in a Kullback-Leibler sense) of the underlying distribution of collision-free trajectories for a given environment. In implementation, cross-entropy method adapted for motion planning starts with drawing K sample trajectories from the GP defined in (3.1), where the mean $\boldsymbol{\mu}$ is initialized as a constant-velocity straight line in configuration space and covariance matrix $\mathcal{K}$ arises from the exactly sparse precision matrix defined in (3.13). We sample trajectories from the GP with (4.9). Subsequently, for each sampled trajectory we evaluate the cost using the aforementioned hinge loss with a precomputed signed distance field defined in (3.22), with the cost of k-th trajectory being $\gamma^k = f[\boldsymbol{\theta}^k(t)]$. From the evaluated

K trajectories, we pick the M best ones according to the optimization criteria, i.e. ones with the lowest cost $f[\boldsymbol{\theta}(t)]$. In a cross-entropy optimization framework, this can be seen as setting the indicator function threshold as $f[\boldsymbol{\theta}] \leq \gamma^M$. Using the sampled trajectories and evaluated costs, the GP parameters $\boldsymbol{\mu}^*, \boldsymbol{\mathcal{K}}^*$ for the next iteration can be computed with (4.14). For Gaussian sampling distributions, it was shown that the problem in (4.14) admits a closed-form solution given by the sample (empirical) mean and covariance [118, 79].

4.3.3 *Updating the mean and the covariance*

We modify the cross-entropy optimization method described in previous section by modifying the indicator function in (4.12). A similar modification was done in [118]. After sampling, the cross-entropy method takes into account M best trajectories according to the computed cost and their contribution to the new mean and covariance is equal regardless of their cost. In motion planning context, if we pick M best trajectory samples in a given highly cluttered environment and one of them has significantly lower cost than others (i.e. it is closer to being collision-free), the computed mean will average it out together with the other $M-1$ and valuable information will be lost. Since we try to find collision-free trajectories, it makes sense to value more the trajectory with the smallest cost. Therefore, we introduce an alternative, *soft* indicator function

$$I_{\{f[\boldsymbol{\theta}]\leq\gamma\}} = \begin{cases} 1/f[\boldsymbol{\theta}] & \text{if } f[\boldsymbol{\theta}] \leq \gamma \\ 0 & \text{if } f[\boldsymbol{\theta}] > \gamma \end{cases}. \tag{4.15}$$

With the introduced alternative indicator function, trajectories with lower cost posses a higher *indicator value*. The solution to the problem in (4.14) then corresponds to the weighted sample mean and covariance [118]. The cost-weighted arithmetic sample mean of the M best trajectories forms a new GP mean $\boldsymbol{\mu}$ for next iteration

$$\boldsymbol{\mu}(t) = \sum_{m=1}^{M} \lambda_m \boldsymbol{\theta}^m(t) \tag{4.16}$$

where weights λ_m correspond to the normalized *indicator values* of given trajectories

$$\lambda_m = \frac{1/f[\boldsymbol{\theta}^m(t)]}{\sum_{m=1}^{M} 1/f[\boldsymbol{\theta}^m(t)]}. \tag{4.17}$$

When computing the GP covariance $\boldsymbol{\mathcal{K}}$ or precision $\boldsymbol{\mathcal{K}}^{-1}$ matrix for the next iteration from the provided M best trajectories, we need to take into account the pertaining sparse structure of the precision matrix and the underlying motion model. Since trajectories are sampled at discrete time instances, we only have access to the support states $\boldsymbol{\theta}_i$. A discrete version of the LTV-SDE in (3.5) is defined as

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\theta}_i + \boldsymbol{\nu}_{i,i+1} + \boldsymbol{w}_{i,i+1}, \tag{4.18}$$

where

$$\boldsymbol{w}_{i,i+1} \sim \mathcal{N}(0, \boldsymbol{Q}_{i,i+1}). \tag{4.19}$$

Mean trajectory of the discrete LTV-SDE in (4.18) can be written as

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\mu}_i + \boldsymbol{v}_{i,i+1}. \tag{4.20}$$

Now we can write exogenous input $\boldsymbol{v}_{i,i+1}$ as

$$\boldsymbol{v}_{i,i+1} = \boldsymbol{\mu}_{i+1} - \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\mu}_i, \tag{4.21}$$

and after substituting back to (4.18) get the residual

$$\boldsymbol{w}^m_{i,i+1} = \boldsymbol{\theta}^m_{i+1} - \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\theta}^m_i - \boldsymbol{\mu}_{i+1} + \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\mu}_i. \tag{4.22}$$

The weighted sample covariance matrix between states i and $i+1$ can be estimated using the calculated mean and sampled trajectories

$$\boldsymbol{Q}^{est}_{i,i+1} = \mathbb{E}[\boldsymbol{w}_{i,i+1}\boldsymbol{w}^T_{i,i+1}] \tag{4.23}$$

$$= \sum_{m=1}^{M} \lambda_m \boldsymbol{w}^m_{i,i+1}\boldsymbol{w}^{mT}_{i,i+1}. \tag{4.24}$$

By calculating $\boldsymbol{Q}^{est}_{i,i+1}$ with (4.24), we implicitly solved the integral in (3.12), determining $Q_c(t)$ in the process. After calculating the covariance estimates $\boldsymbol{Q}^{est}_{i,i+1}$ for each pair of states, we can construct the matrix $\widetilde{\boldsymbol{Q}}^{est}$ similarly to (3.11)

$$\widetilde{\boldsymbol{Q}}^{\boldsymbol{est}} = \mathrm{diag}(\mathcal{K}_0, \boldsymbol{Q}^{est}_{0,1}, ..., \boldsymbol{Q}^{est}_{N-1,N}), \tag{4.25}$$

and calculate the precision matrix similarly to (3.13)

$$[\mathcal{K}^{est}]^{-1} = \boldsymbol{G}^T[\boldsymbol{Q}^{est}]^{-1}\boldsymbol{G}, \tag{4.26}$$

thus retaining the initial and goal covariances $\mathcal{K}_0$, $\mathcal{K}_N$ in order to keep the start and goal states fixed and preserving the block-tridiagonal structure.

The constant-velocity model that underpins our GP remains intact through iterations, meaning that smoothness is retained while searching for collision-free trajectory. The shape of the estimated covariance matrix determines relatively how the trajectories sampled in next iteration will deviate from the mean, e.g., in parts of the environment with tight spaces, trajectories will only slightly deviate from the mean, while the parts of trajectories that pass through open space will deviate more. While the described covariance estimation process correctly captures the shape of the covariance, it might have trouble capturing the correct value. This problem stems from normalizing the *indicator value* - we lose information about absolute values of the cost i.e. how close our trajectories actually are to being collision free. This means that such method will have similar covariance values for problems where sampled trajectories are highly in collision and problems where only small parts of sampled trajectories are in collision. Intuitively, we know that if most parts of trajectories are in collision we want larger values of covariance to reinforce exploration, while if only small parts are in collision smaller covariance should be enough to explore around the area locally and find a collision-free trajectory. To obviate this problem, we opt to scale the estimated covariance with the cost value of the calculated mean

$$\mathcal{K} = \alpha f[\boldsymbol{\mu}(\boldsymbol{t})]\mathcal{K}^{est}, \tag{4.27}$$

Algorithm 1 Stochastic Trajectory Optimization with GPs

Input: Start and goal states $\boldsymbol{\theta}_0$, $\boldsymbol{\theta}_N$, a state-dependent cost function $f[\boldsymbol{\theta}^k(t)]$
Precompute: Initial mean $\boldsymbol{\mu}$ and covariance $\mathcal{K}$

```
for 1 ... N_iter do
    for 1 ... K do
        Sample trajectory θ^k(t) ~ GP(μ(t), K(t, t'))
        Evaluate trajectory cost f[θ^k(t)]
        if f[θ^k(t)] = 0 then
            Return collision free trajectory θ^k(t)
        end if
    end for
    From K sampled trajectories take M with lowest cost
    Compute weights λ_m using (4.17)
    Compute new mean μ(t) = Σ_{m=1}^{M} [λ_m θ^m(t)]
    Compute new precision matrix using (4.24)-(4.28)
end for
```

or equivalently scale the precision matrix

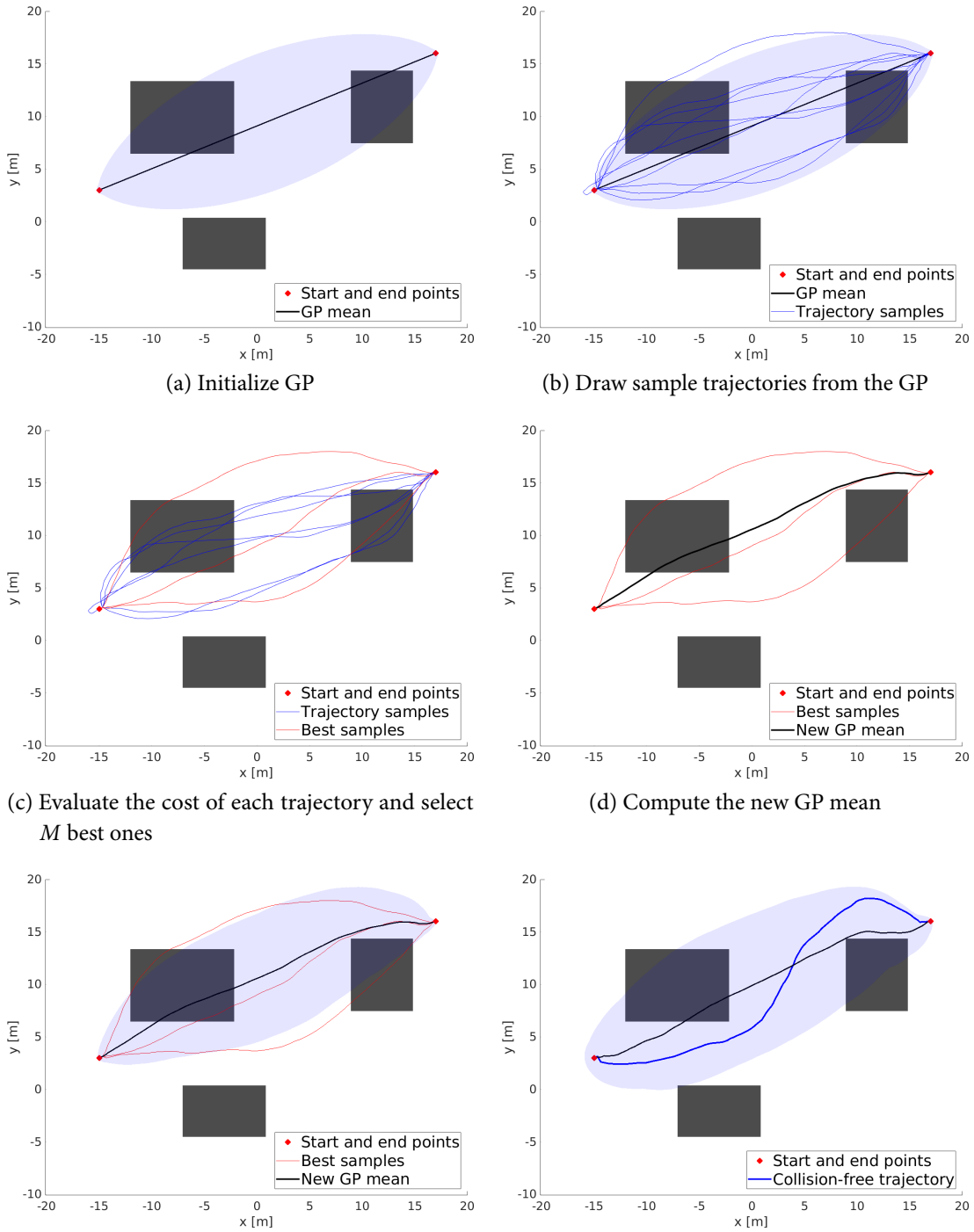
$$\mathcal{K}^{-1} = \frac{1}{\alpha f[\boldsymbol{\mu}(t)]}[\mathcal{K}^{est}]^{-1}, \tag{4.28}$$

where α is a scalar parameter that regulates the sensitivity of the covariance or precision matrix scaling. Larger mean cost imposes larger covariance values and thus encourages exploration. The current mean can be used in the next iteration as one of the sampled trajectories so that computing its cost does not introduce additional computational burden.

To summarize, we start by sampling K trajectories from a GP initialized with a constant-velocity straight line. For each sampled trajectory we compute its associated cost and select M best ones according to the cost. We then form the new GP mean and precision matrix with (4.16) and (4.28). This process is repeated until a collision-free trajectory is found. The described method is summarized in Algorithm 1 and illustrated in Figure 4.2.

4.3.4 *Computational Efficiency Remarks*

In order to thoroughly explore the environment, our approach requires cost evaluation for relatively many drawn trajectory samples, which naturally leads to a slower computation than the state-of-the-art gradient based methods. However, due to the fact that cost evaluation for each trajectory is independent, the inner for loop in the proposed Algorithm 1 can be parallelized with computational efficiency scaling linearly with the number of processing cores. In our implementation, we exploit this property and parallelize the inner loop on 4 processing cores. A GPU implementation presents an interesting possibility that would allow sampling and cost evaluation for a huge number of trajectories, leading to fast environment exploration and discovering optimal trajectories allowing real-time replanning in dynamic environments [119].



(a) Initialize GP

(b) Draw sample trajectories from the GP

(c) Evaluate the cost of each trajectory and select M best ones

(d) Compute the new GP mean

(e) Compute the new GP covariance

(f) Repeat until a collision-free sample trajectory is found

Figure 4.2: An example illustrating the steps of the proposed algorithm in a 2D environment with three obstacles. In this simple example the number of sampled trajectories is set as $K = 10$ and the number of chosen best ones $M = 3$.

We use GP interpolation for dense collision checking, similarly to [59]. Since trajectory *support states* are temporally equidistant and each sampled trajectory is drawn from the same GP, matrices $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ in (3.15) and (3.16) can be precomputed, instead of computing them each time interpolation is needed. This provides another significant increase in the

proposed method's computational efficiency.

4.4 RESULTS

In this section, we demonstrate the performance of the proposed method. We conducted three simulation experiments to evaluate the proposed method. The method was quantitatively tested in two simulated benchmarks and compared with the state-of-the-art trajectory optimization techniques GPMP2 [59] and STOMP [71]. We also demonstrate our method's capabilities in a qualitative test. In Section 4.4.1, we demonstrate the improvement of the proposed method over GPMP2 with random restarts and STOMP in solving a 2D maze, which is a suitable benchmark for an optimization-based planner effectiveness at finding a collision-free solution in a haystack of local minima. This experiment aims to show benefits of the proposed stochastic method, which allows for better exploration in comparison to gradient-based methods. In Section 4.4.2, we show the improvement of the proposed method over prior techniques in finding a collision-free trajectory for a 7 DoF manipulator in cluttered environment. This experiment attests to the benefits of the proposed heteroscedastic GP prior since the environment was set up so that obstacles are placed near the start and goal state. In Section 4.4.3, we qualitatively demonstrate the ability of the proposed method to find a collision-free trajectory for a 10 DoF mobile manipulator in a cluttered environment. This experiment aims to show general capabilities of the proposed method in motion planning for high DoF robots in complex environments.

In all three experiments, our method was always initialized with a constant-velocity straight line trajectory in the configuration space. Signed distance parameter was set as $\varepsilon = 0.1\,\text{m}$. We keep the parameter ε, which indicates *safety distance*, relatively small since our primary goal is collision avoidance. Larger values of ε sometimes tend to penalize trajectories that are collision-free but close to obstacles more than trajectories that are in collision at one point but far from obstacles for the most part. The covariance scaling parameter was set as $\alpha = 0.5$, for which we empirically determined that it provides good balance between thorough exploration and fast convergence. In the two conducted benchmarks, along with testing the proposed method we also tested the variant without the proposed covariance estimation procedure, keeping the covariance matrix $\mathcal{K}$ unchanged through iterations, as was done in [111]. The unchanged covariance matrix allows for exhaustive exploration around the mean in each iteration. Changing only mean $\boldsymbol{\mu}$ while keeping the covariance matrix $\mathcal{K}$ unchanged is permitted in the proposed GP framework described in Section 3.1, as change in $\boldsymbol{\mu}$ can be attributed to some implicitly imposed exogenous input $\boldsymbol{\nu}(t)$ which does not impact covariance.

For GPMP2, we used a straight line initialization as a baseline, and in our experiments we designate to this model as *line*. Since GPMP2 always converges very quickly, but often fails in cluttered environments due to infeasible local minima, we also employed random restarts, which is a commonly used method to tackle the local minima problems in gradient-based trajectory optimization methods [47]. In this technique, the optimizer is first initialized with a straight-line and then, on failure, re-initialized with a random trajectory. Our implementation samples the random restart trajectory from a homoscedastic GP, similarly to [60]. We designate to this model as *rr*. For STOMP, we also used a constant-velocity

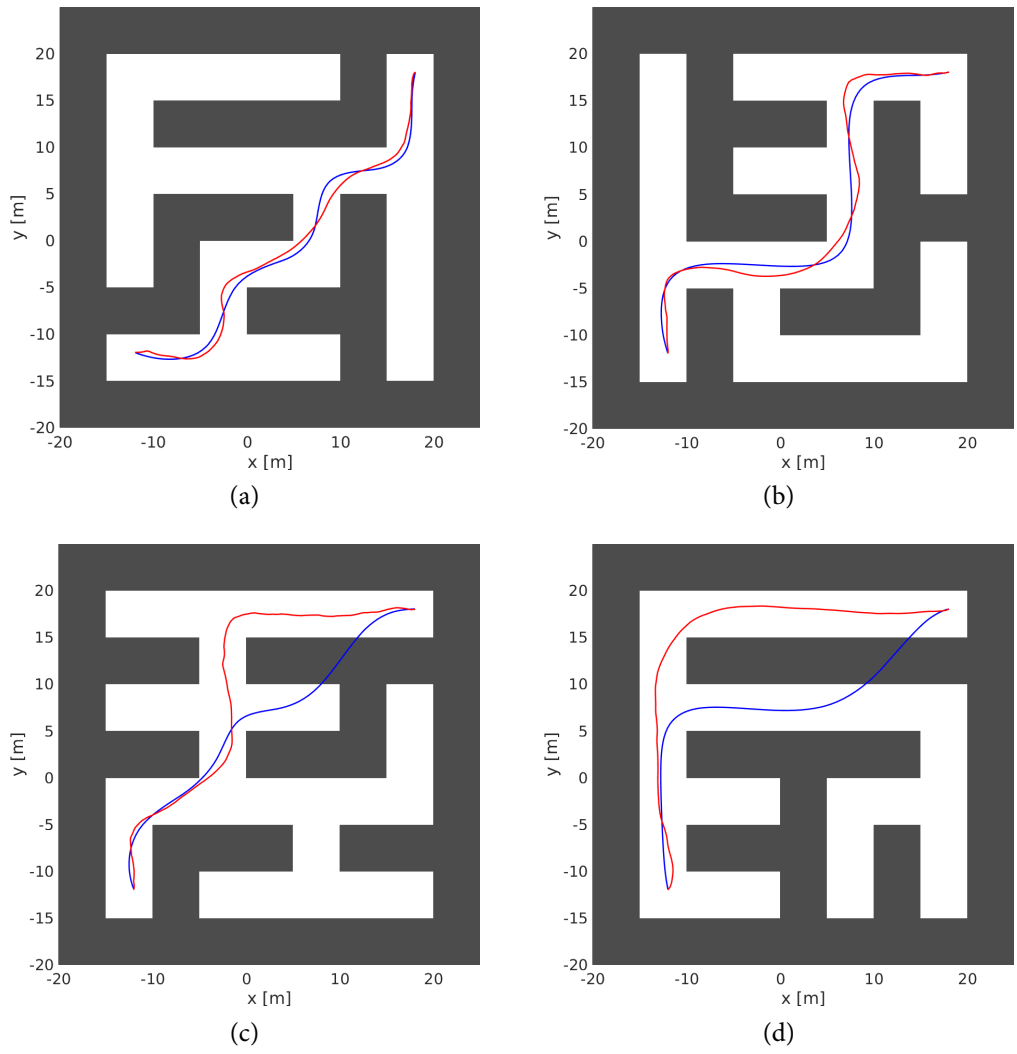


Figure 4.3: Examples of 4 × 4 mazes where the proposed approach finds a collision-free solution. Blue line represents trajectory found by GPMP2, while red line represents trajectory found by our method. Slight undulation of trajectories obtained by the proposed method is due to the criterion of finding a collision-free trajectory, unlike the GPMP2 criterion which explicitly encourages smoothness.

straight line initialization. The parameter that determines the number of noisy trajectories generated at each STOMP iteration was set to $K = 5$, which was demonstrated to achieve good performance in prior work [71].

We used the GPMP2 C++ library [59, 120] and its respective MATLAB toolbox based on the GTSAM C++ library [121], while we used our own implementation of STOMP. All experiments were performed on a system with a 2.8-GHz Intel Core i7-7700HQ processor and 16 GB of RAM.

4.4.1 *The Maze Benchmark*

The maze benchmark, appropriate for quantitative evaluation, consisted of 1000 synthetic environments created by the Wilson's algorithm [122], which generates uniformly sampled mazes with a single solution (i.e. perfect mazes). Mazes were generated on grids with sizes

of 3×3, 4×4 and 5×5 and afterwards inflated to realistic dimensions. While the 2D maze problem is generally suitable for grid-based or sampling-based motion planning approaches which achieve 100% success rate, it can be used to measure an optimization-based planner's effectiveness at finding the unique collision-free solution in a cluttered environment.

For each maze environment, we plan motion for a 2D holonomic circular robot with the radius of 0.5 m. For both variants of our method, we chose the number of sampled trajectories $K \in [200, 400]$, while the number of best trajectories chosen for the weighted mean in each iteration was set to $M = 3$. Although these parameters may seem disproportionate, choosing a large K ensures exhaustive exploration, while a small M induces drastic changes in the GP mean $\boldsymbol{\mu}$ between iterations, which helps in finding the solution faster in complex environments. We set the total trajectory time (i.e. the timespan in which the robot moves from the start to goal state) to $t_{\text{total}} = 20$ s, while time-varying covariance matrix of the white noise governing the heteroscedastic GP was calculated as $\boldsymbol{Q}_c(t) = (t - \frac{t_{\text{total}}}{2})^2 \boldsymbol{I}$, which generates a parabola with its vertex at the midpoint of the trajectory. For GPMP2 we set $\epsilon = 2$ and $Q_c = 1$ according to the Matlab toolbox 2D example. To achieve the best performance on our benchmark, we tuned the GPMP2 algorithm, testing it with the parameter σ_{obs} in range $[0.01, 0.1]$ with increments of 0.01 and in range $[0.1, 1]$ with increments of 0.1. For the 3x3 benchmark, reported GPMP2 performance was achieved with $\sigma_{obs} = 0.3$, while for the 4x4 and 5x5 benchmark it was achieved with $\sigma_{obs} = 0.1$.

For the proposed method and GPMP2, the trajectory was parameterized with $N = 10$ *support states* and 5 interpolation steps in-between for which the trajectory cost is evaluated. While larger number of *support states* N would likely lead to somewhat better performance, it would negatively impact computation time. GPMP2 demonstrated that using less *support states* with more interpolated states inbetween provides optimal performance and this translates to the proposed method since we utilize similar underlying GP framework. For STOMP, the trajectory was parameterized with $N = 50$ discrete states. We set the maximum runtime for our algorithm, STOMP and random restarts as $t_{\text{max}} = 1$ s, with one exception where we set $t_{\text{max}} = 2$ s in order to investigate the ability of our algorithm to find solutions given more time. We measure the number of mazes solved (success rate) and the execution time. Figure 4.3 depicts examples of four 4x4 mazes with obtained trajectories with the proposed method and GPMP2. We do not depict trajectories obtained with STOMP since it consistently achieved both the worst success rate and execution time in the maze experiment. Table 4.1 shows the experiment results in detail.

For every maze complexity level, the proposed method with covariance estimation achieved better results (higher success rate and lower execution time) than the variant without covariance estimation when the maximum algorithm runtime was set to $t_{\text{max}} = 1$ s. While calculating the new covariance matrix in each iteration takes additional computation time, the method converges faster, i.e. it needs less iterations to find a collision-free solution. On average, it required approximately 2.5× less iterations. If a trajectory that is close to being collision-free is found, the estimated covariance is rather small and thus the focus shifts to exploring the area around that trajectory locally. Without covariance estimation, the initial covariance carries exhaustive exploration of the space in each iteration, which can be sample inefficient since the algorithm will sample trajectories through the whole environment even when the mean is close to a collision-free trajectory. In other words, covariance

estimation reduces the search space and thus leads to faster convergence. However, when given additional computation time (algorithm runtime set to $t_{\max} = 2\,\mathrm{s}$), the proposed approach without covariance estimation achieves marginally better success rate in all three maze types. This stems from the aforementioned fact that the covariance estimation reduces the search space which can sometimes lead to exploring locally around local minima. The variant without covariance estimation is more reliant on "lucky" sampling, it possesses better ability to stumble upon a collision-free trajectory given enough time. It is also worth noting that, through covariance estimation, the proposed method approximates the covariance of the underlying distribution of collision-free trajectories. It is thus less reliant on finely tuned initialization than the variant without covariance estimation, where providing too large initial covariance matrix $\mathcal{K}$ leads to degraded performance. Without covariance estimation, sampling $K = 200$ and $K = 400$ trajectories in each iteration lead to similar results; however, with covariance estimation, sampling $K = 200$ trajectories proved to find the solution significantly faster than sampling $K = 400$ trajectories. This attests to the benefits of covariance estimation where the algorithm adapts to the given environment through iterations and is more sample-efficient.

STOMP achieved both the worst success rate and execution time for every maze complexity level. While STOMP does not require gradient and samples noisy trajectories, it tries to combine sampled trajectories to produce an update rule that produces smooth trajectories, but often fails to explore the environment. Unlike the proposed method, if one of sampled noisy trajectories is collision-free, it is not used as a solution, but utilized to produce an update to the mean trajectory. In a complex environment such as maze, this often leads to the resulting trajectory being in collision, even if a collision-free trajectory was found during the sampling process. While the GPMP2 without random restarts has almost an order of magnitude faster execution time than every other tested method, it has significantly worse success rate than random restarts and the proposed method for every maze complexity level. This was expected, as it is a classical gradient-based trajectory optimization algorithm meaning that it is prone to local minima which are omnipresent in a maze environment. The proposed approach also outperformed the GPMP2 with random restarts for every maze complexity level. For the mazes created from a 3×3 grid, our algorithm managed similarly high success rate with twice as fast computation. For the mazes created from a 4×4 grid our algorithm vastly outperformed random restarts, having notably higher success rate with similar reported times. The most complex mazes created from a 5×5 grid demonstrated the inability of gradient-based methods to find solutions in environments plagued with multitude of local minima, while the proposed method achieved moderate success. Note that the reported execution time for our algorithm is the actual time it took to compute on all cores, and not the sum of times over them.

4.4.2 *The Robot Arm Planning Benchmark*

The robot arm planning benchmark consisted of a simulated WAM robotic arm in an environment featuring a table and a drawer cabinet. We conducted 20 unique planning experiments, all with different start and goal states with starting points being under the table and end states being inside the cabinet. This set of problems is not particularly difficult

Table 4.1: Success rate (%) / average execution time (ms) on maze and robot arm planning benchmarks.

	Proposed approach with covariance estimation			Proposed approach without covariance estimation			GPMP2		STOMP
Maze	$K = 400$, $t_{max} = 2\,s$	$K = 400$, $t_{max} = 1\,s$	$K = 200$, $t_{max} = 1\,s$	$K = 400$, $t_{max} = 2\,s$	$K = 400$, $t_{max} = 1\,s$	$K = 200$, $t_{max} = 1\,s$	rr	line	
3x3	92 / 107	91.5 / 97	90.2 / 55	95.2 / 197	92.9 / 171	89.0 / 160	89.2 / 96	55.3 / 28	51.4 / 600
4x4	72.4 / 240	70.9 / 209	66.9 / 128	79.1 / 489	66.9 / 324	61.8 / 297	58.1 / 298	20.5 / 40	9.5 / 786
5x5	42.7 / 520	37.3 / 381	37.4 / 299	43.8 / 858	26.7 / 493	26.3 / 429	19.8 / 273	9.6 / 36	0.2 / 742
Arm	Heteroscedastic $K = 400$, $t_{max} = 1\,s$	Homoscedastic $K = 400$, $t_{max} = 1\,s$		Heteroscedastic $K = 400$, $t_{max} = 1\,s$	Homoscedastic $K = 400$, $t_{max} = 1\,s$		rr	line	
	100 / 494	90 / 680		100 / 458	75 / 569		85 / 372	65 / 84	30 / 832

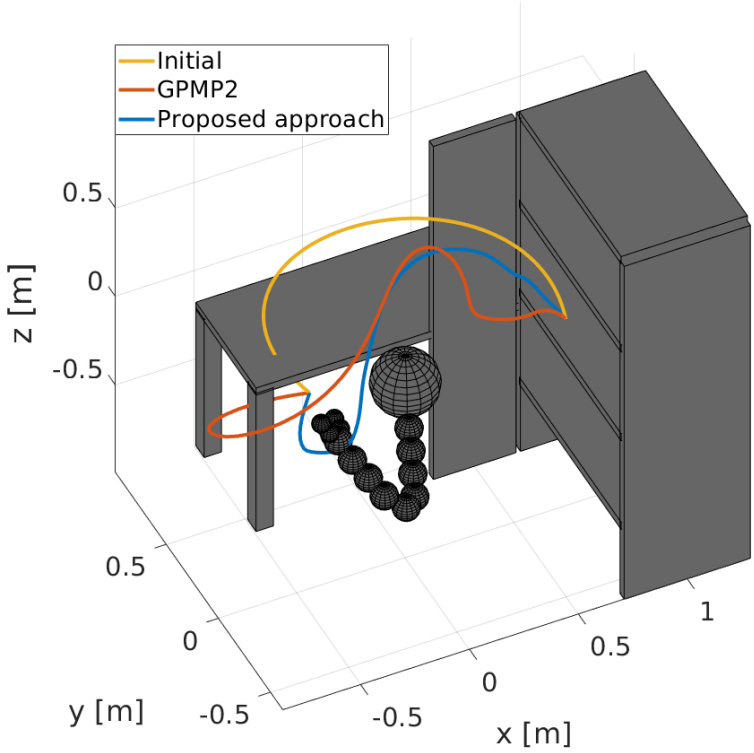


Figure 4.4: A simulated WAM robotic arm in an environment featuring a table and a drawer cabinet. Plotted lines depict the end effector trajectories. This is an example where the proposed approach finds a collision free solution, while GPMP2 converges to the infeasible local minimum. Initial straight-line trajectory in configuration space is also shown.

since most of the states are initially collision free; however, it was set up to accentuate the proneness of the homoscedastic GP motion planning methods to get stuck in local minima near start or goal states.

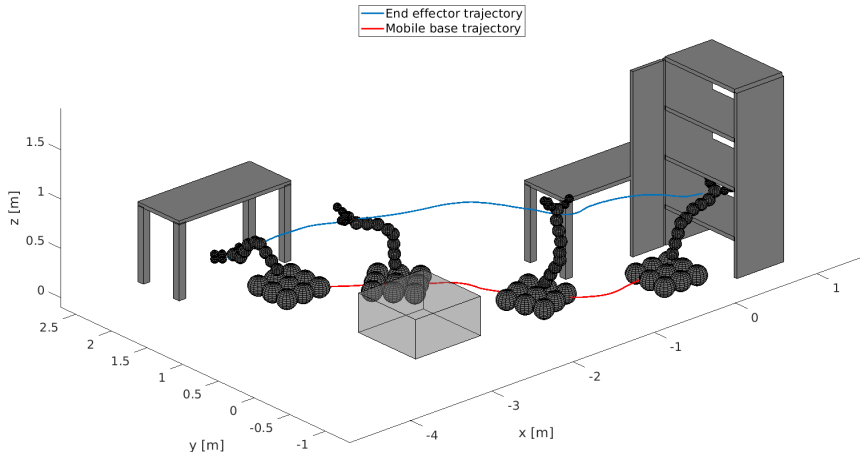
In this benchmark we tested both variants of the proposed method with heteroscedastic and homoscedastic GP priors in order to demonstrate the benefits of heteroscedasticity. For both variants of our method, we chose the number of sampled trajectories $K = 400$, while the number of best trajectories chosen for the weighted mean in each iteration was set as $M = 3$. We set the total trajectory time to $t_{\text{total}} = 20\,s$. For homoscedastic cases we chose $\boldsymbol{Q}_c = 0.02\boldsymbol{I}$, while for a heteroscedastic GPs we calculated $\boldsymbol{Q}_c(t) = 0.01(t - \frac{t_{\text{total}}}{2})^2\boldsymbol{I}$. The optimal coefficients of $\boldsymbol{Q}_c$ were found empirically, as is the case in other applications of the GP trajectory representation [58, 63]. We utilized grid search to tune the GPMP2 algorithm,

searching for both ϵ in range $[0.01, 0.1]$ with increments of 0.1 and σ_{obs} in range $[0.1, 1]$ with increments of 0.1. The reported GPMP2 performance on this benchmark was achieved with parameters $\epsilon = 0.02$ and $\sigma_{obs} = 0.2$. For the proposed method and GPMP2, the trajectory was parameterized with $N = 10$ equidistant *support states* and 10 interpolation steps in-between for which the trajectory cost was evaluated. For STOMP, the trajectory was parameterized with $N = 100$ discrete states. We again set the fixed time budget for our algorithm, STOMP and random restarts as $t_{\max} = 1\,\text{s}$ and measured the success rate and the execution time. Due to stochastic nature of the results, we repeated every planning experiment with our method and with STOMP 20 times to correctly assess the required computation time. We measured the average success rate and the average algorithm execution time. The results of the experiment are shown in Table 4.1, where we can see that while the baseline GPMP2 often fails, the stochasticity introduced by random restarts helps in achieving higher success rates. An example of a case where our method finds a solution and GPMP2 fails is depicted in Figure 4.4. Interestingly, when the proposed approach with covariance estimation was initialized with a homoscedastic GP, the covariance estimation guided the GP towards more exploration near the start and end points, leading to better success rate than a homoscedastic GP without covariance estimation, albeit requiring additional computational time to converge. Both variants of the proposed method with heteroscedastic GPs achieved a perfect score within the fixed time budget, thus demonstrating the advantage of the proposed heteroscedastic prior for solving problem instances with obstacles near the start or end points of the robot trajectory.

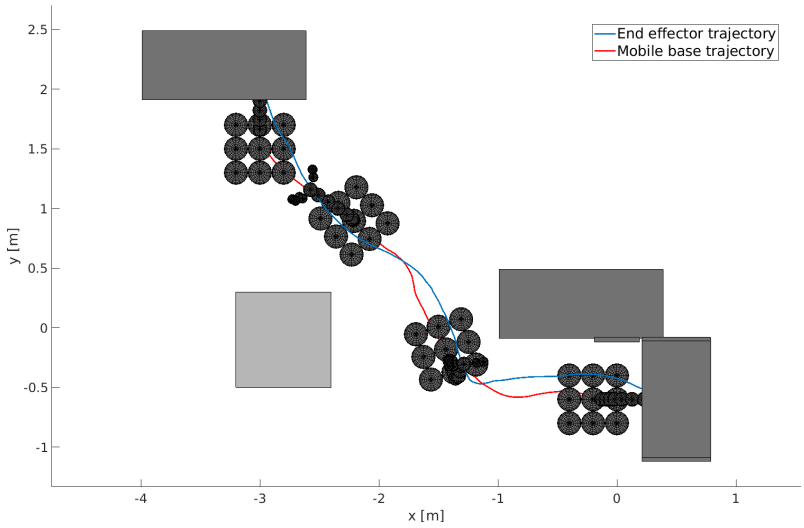
4.4.3 *Mobile Manipulator Trajectory Planning*

To demonstrate the ability of the proposed method to find a collision-free trajectory for high DoF robots in complex environments, we conducted a mobile manipulator trajectory planning experiment. The simulated mobile manipulator consisted of a simulated WAM robotic arm mounted on top of an omnidirectional platform in an environment featuring two tables, a drawer cabinet and a generic static obstacle. The system has 10 DoF corresponding to mobile platform's 3 DoF pose and robot arm's 7 joint angles. We planned the trajectory of the whole system, simultaneously planning the motion of the arm and the platform.

We chose the number of sampled trajectories $K = 200$, while the number of best trajectories chosen for the weighted mean in each iteration was set as $M = 3$. We set the total trajectory time $t_{\text{total}} = 10\,\text{s}$. We modelled the time-varying covariance matrix of the white noise governing the heteroscedastic GP as an anisotropic diagonal matrix. The elements of the matrix associated with the omnidirectional platform were calculated as $\boldsymbol{Q}_c(t) = 0.1(t - \frac{t_{\text{total}}}{2})^2$, while the elements associated with the robot arm states were calculated as in the robot arm planning benchmark $\boldsymbol{Q}_c(t) = 0.01(t - \frac{t_{\text{total}}}{2})^2$. The coefficients of $\boldsymbol{Q}_c(t)$ were found empirically. The noise induced for a 2D mobile platform was much smaller in comparison to the 2D maze benchmark since start and end points of this problem were much closer to each other and the environment required less exploration. In such case, smaller $\boldsymbol{Q}_c(t)$ should still lead to a collision-free solution, but the resulting trajectory will be smoother. The trajectory was parameterized with $N = 20$ equidistant *support states* and 10 interpolation steps in-between for which the trajectory cost was evaluated. To assess the



(a) 3d view



(b) 2d view

Figure 4.5: An example of a trajectory obtained with the proposed method on a simulated mobile manipulator in a static environment.

required computation time, we repeated this experiment 30 times. Every take resulted in successfully finding a collision-free trajectory and the average algorithm execution time was 788 ms, while the worst case computation time was 1362 ms. The result of this experiment is shown in Figure 5.7. The experiment demonstrated the ability of the proposed approach to find collision-free trajectories for a mobile manipulator in a cluttered environment.

4.5 SUMMARY

In this chapter, we tackled the problem of using trajectory optimization for achieving successful high-dimensional motion planning in cluttered environments. We have presented a stochastic optimization method for trajectory planning, representing robot trajectories as samples from a continuous-time GP. The continuous time representation leads to smooth trajectories that can be queried at any time of interest, while introducing the heteroscedasticity of the underlying GP enabled us to generate trajectory priors better suited for collision avoidance in motion planning problems. We derived a cross-entropy based derivative-free stochastic optimization method that estimates the distribution of collision-free trajectories for a given environment, utilizing importance sampling in order to contend with the local minima problem present in trajectory optimization methods. Due to the sparsity of the employed GP framework, we were able to efficiently estimate the GP covariance at each iteration. We evaluated our method on three simulation scenarios: a maze benchmark, a 7 DoF robot arm planning benchmark and a 10 DoF mobile manipulator trajectory planning example. In a maze benchmark, the proposed method yielded a more thorough exploration of the solution space in complex environments than GPMP2 and STOMP, while having comparable execution time. The robot arm planning benchmark demonstrated the benefits of heteroscedastic GP, while the mobile manipulator planning experiments showed that the proposed method is capable of efficiently tackling high-dimensional trajectory planning problems.

5

Multiple-criteria motion planning

This chapter presents a novel approach to multiple-criteria high-dimensional robot motion planning. The proposed motion planning method represents trajectories as samples from a mixture of continuous-time Gaussian processes (MGP) and employs stochastic optimization in order to update the MGP parameters in a cost-minimizing manner. The contributions of the proposed trajectory optimization method arise from the introduced mixture representation and stochastic gradient estimation, dominantly enabling better exploration of the trajectory space and including non-differentiable optimizing costs, which is not supported by most of the state-of-the-art methods. We evaluated the proposed method in multiple simulation benchmarks featuring 7 DoF robot arms and a 10 DoF mobile manipulator. Besides collision avoidance, we evaluated performance in torque minimization and singularity avoidance. We also conducted laboratory experiments with a 14 DoF dual arm robot Rethink Baxter, aiming to corroborate performance of the proposed method in a real-world setting. The experimental results demonstrated that the proposed method achieves higher success rate than several state of the art methods, while the advantages stemming from MGPs and stochastic optimization, like trajectory smoothness, support of non-differentiable cost functions, multiple trajectory solutions, and the ability to tackle high-dimensional planning problems, are inherently kept.

5.1 Introduction

While the current state of the art demonstrated impressive results in collision avoidance for manipulation planning, there are several key areas which they often neglect. Sampling-based approaches often disregard trajectory smoothness resulting in suboptimal motion which causes mechanical stress on robot's hardware. Furthermore, including various task-space or kinodynamic constraints or incorporating singularity avoidance is often difficult to achieve while keeping the method tractable in practice for high-dimensional configuration spaces [123]. Trajectory optimization methods, on the other hand, offer trajectory smoothness, but at the cost of performance in collision avoidance, as both the smoothness cost and the collision avoidance cost are summed together into a single optimization criteria. Moreover, most of the state-of-the-art trajectory optimization methods [47, 45, 59] have problems incorporating joint and velocity limits and require first-order information about the included costs, limiting the inclusion of cost functions to differentiable ones.

The approach that we propose in this chapter relies on mixtures of continuous-time

Gaussian processes (MGPs) for trajectory representation and a trajectory optimization method based on stochastic gradient estimation. The main advantages of the proposed approach arise from the introduced mixture representation and stochastic gradient estimation that in conjunction result with smooth trajectories covering large parts of the trajectory space, and enable optimization of non-differentiable costs. The proposed method can be viewed as an instance of trajectory optimization that also draws a connection to sampling-based approaches. It relies on the interplay between sampling random trajectories and a gradient update rule, thus resulting with high degree of exploration while producing smooth solutions in a computationally efficient manner. Opposed to the method proposed in Chapter 5, the proposed method better covers the configuration space with regards to criteria such as smoothness and torques while performing similarly in collision avoidance. Furthermore, while the method in Chapter 5 employs a cross-entropy-based optimization, the herein proposed method utilizes an update rule similar to gradient descent but employing a stochastic gradient estimation technique. Due to this update rule, it is significantly more sample efficient than the method in Chapter 5 and thus requires less computational resources in most practical cases. It is also able to efficiently optimize multiple different optimization criteria.

The main features of the proposed method are:

(i) continuous-time GPs as mixture components lead to solution trajectories that can be queried at any time of interest and do not require post-processing,

(ii) the proposed MGP initializations cover large parts of trajectory space alleviating the local minima problem while retaining smoothness,

(iii) the proposed stochastic gradient estimation respects joint and velocity limits and allows for optimizing costs that are non-differentiable,

(iv) the MGP representation allows finding multiple solution trajectories to a given problem.

These features bring advantages to our method and make it unique since none of the other existing trajectory optimization methods possess all of them. We evaluated our method in multiple simulation benchmarks featuring 7 DoF robot arms and a 10 DoF mobile manipulator. We also conducted a real-world experiment with a 14 DoF dual-arm robot Rethink Baxter. The results demonstrated that the proposed method achieves a higher success rate than several state-of-the-art methods, while inherently possessing the advantages stemming from MGPs and stochastic optimization, like trajectory smoothness, support of non-differentiable costs, multiple trajectory solutions, and the ability to tackle high-dimensional planning problems. The proposed method is accompanied by our two research papers; in [124] the main theoretical developments were presented, while in [125] we incorporate singularity avoidance in trajectory optimization.

5.2 Background: Trajectory optimization as probabilistic inference

In this section, we present the general framework for formulating trajectory optimization problem as probabilistic inference, relying on the continuous-time GP suitable for motion

planning presented in Section 3.1. This formulation follows developments from [46] and can be exploited for efficient trajectory optimization. To formulate motion planning as probabilistic inference, we cast the motion planning problem as searching for the *maximum a posteriori* (MAP) continuous-time trajectory starting from a prior distribution on the space of trajectories and an arbitrary likelihood function. More formally, we look for the posterior density of $\boldsymbol{\theta}$ given a collection of events $\boldsymbol{e}$ which can be calculated via Bayes' rule

$$p(\boldsymbol{\theta}|\boldsymbol{e}) = \frac{p(\boldsymbol{\theta})p(\boldsymbol{e}|\boldsymbol{\theta})}{p(\boldsymbol{e})} \propto p(\boldsymbol{\theta})p(\boldsymbol{e}|\boldsymbol{\theta}), \tag{5.1}$$

where $p(\boldsymbol{\theta})$ represents the prior on trajectory $\boldsymbol{\theta}$ and $p(\boldsymbol{e}|\boldsymbol{\theta})$ is the likelihood of an event $\boldsymbol{e}$ given trajectory $\boldsymbol{\theta}$.

In previous works [59, 70], a prior distribution on the space of trajectories, whose role is to encourage the smoothness of the trajectory, is a GP defined by the mean $\boldsymbol{\mu}$ and covariance $\mathcal{K}$ as in (3.4), while the likelihood function, whose role is to encourage posterior trajectories that are collision-free, is commonly defined as a distribution in the exponential family

$$L(\boldsymbol{\theta}; \mathbf{e}) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{h}(\boldsymbol{\theta}, \mathbf{e})\|^2_{\boldsymbol{\Sigma}}\right\}, \tag{5.2}$$

where $\boldsymbol{h}(\boldsymbol{\theta}, \mathbf{e})$ is an arbitrary vector-valued cost function with covariance matrix $\boldsymbol{\Sigma}$.

The optimal trajectory $\boldsymbol{\theta}^*$ is then found by minimizing the negative logarithm of the posterior $p(\boldsymbol{\theta}|\boldsymbol{e})$ as follows [46]

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left\{\text{-log}\left(p(\boldsymbol{\theta})p(\boldsymbol{e}|\boldsymbol{\theta})\right)\right\} \tag{5.3}$$

$$= \arg\min_{\boldsymbol{\theta}} \left\{-\frac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{\mu}\|^2_{\mathcal{K}} - \frac{1}{2}\|\boldsymbol{h}(\boldsymbol{\theta}, \mathbf{e})\|^2_{\boldsymbol{\Sigma}}\right\}. \tag{5.4}$$

If a prior on the space of trajectories is given as a GP, the negative logarithm reduces the MAP problem to nonlinear least squares that can be solved with iterative optimization techniques such as Gauss-Newton or Levenberg-Marquardt [59].

5.3 Proposed method for trajectory optimization

In this section we derive the proposed method for trajectory optimization, which we dubbed *Mixture of Gaussian Processes for Trajectory Optimization* (MGPTO). First we introduce robot trajectory representations as mixtures of Gaussian processes, then we formulate the trajectory optimization problem with an MGP trajectory prior. Afterwards, we present the procedures for stochastic gradient estimation and for weights update of the MGP. At the end, we present the MGPTO method in the form of an algorithm.

5.3.1 *Robot trajectories as mixtures of Gaussian processes*

A single GP trajectory representation allows searching for a collision-free trajectory in the neighborhood of the mean trajectory, which is often initialized as a constant-velocity straight line, but it does not allow thorough exploration of the environment, which is needed for local minima avoidance. Therefore we consider a continuous-time trajectory as a sample from

an MGP, allowing for broader initialization and thus better coverage of the environment. The MGP model is defined as a linear superposition of multiple GP components

$$p(\boldsymbol{\theta}(t)) = \sum_{m=1}^{M} \pi_m \mathcal{GP}(\boldsymbol{\mu}_m(t), \mathcal{K}_m(t, t')), \tag{5.5}$$

where π_m are the mixing coefficients such that $0 \leq \pi_m \leq 1$, $\sum_{m=1}^{M} \pi_m = 1$. Since our main idea is to represent trajectories as samples from an MGP and employ optimization in order to update the MGP parameters in a cost-minimizing manner, we will need to formulate the MGP in terms of discrete *latent* variables to make this problem soluble. Our development is similar to the Gaussian mixture model widely employed in machine learning [126], where the latent variable, which is usually unknown, represents which Gaussian component generated which observation. In our case the problem is a bit different, since we know in advance which MGP component generated which trajectory.

First, we introduce the latent variable $\boldsymbol{z}$ that is an M-dimensional discrete random variable

$$p(\boldsymbol{z}) \sim \text{Categorical}(\boldsymbol{\pi}), \tag{5.6}$$

where the values of $\boldsymbol{z}$ have a 1-of-M encoding, meaning that only a particular vector element z_m is equal to 1, while all other vector elements are equal to 0. A certain vector element z_m being equal to 1 refers to the m-th mixture component. The joint distribution $p(\boldsymbol{\theta}, \boldsymbol{z})$ can be defined in terms of the marginal and conditional distribution

$$p(\boldsymbol{\theta}, \boldsymbol{z}) = p(\boldsymbol{z}) p(\boldsymbol{\theta}|\boldsymbol{z}). \tag{5.7}$$

The marginal distribution $p(\boldsymbol{z})$ is specified in terms of the aforementioned mixing coefficients $p(z_m = 1) = \pi_m$, while the conditional distribution, given a particular value for $\boldsymbol{z}$, is a GP, $p(\boldsymbol{\theta}|z_m = 1) = \mathcal{GP}(\boldsymbol{\mu}_m(t), \mathcal{K}_m(t, t'))$. Given that, the marginal $p(\boldsymbol{\theta})$ can then be computed by summing over all possible states of $\boldsymbol{z}$

$$p(\boldsymbol{\theta}) = \sum_{\boldsymbol{z}} p(\boldsymbol{z}) p(\boldsymbol{\theta}|\boldsymbol{z}) = \sum_{m=1}^{M} \pi_m \mathcal{GP}(\boldsymbol{\mu}_m(t), \mathcal{K}_m(t, t')). \tag{5.8}$$

Thus the marginal distribution of $\boldsymbol{\theta}$ is an MGP of the form (5.5). We have therefore found an equivalent formulation of the MGP involving an explicit latent variable. This formulation implies that every trajectory $\boldsymbol{\theta}^k$ has a corresponding latent variable $\boldsymbol{z}^k$ that determines from which GP component the pertaining trajectory was generated.

↬ DRAWING TRAJECTORY SAMPLES. To draw a sample from an MGP involving a latent variable, we first sample the latent variable $\boldsymbol{z}$ with the roulette wheel selection where a given mixture weight π_m is the probability of $z_m = 1$. A sample trajectory can then be generated from the mixture component corresponding to the selected latent variable z_m with

$$\boldsymbol{\theta} = \boldsymbol{\mu}_m + \boldsymbol{A}_m \boldsymbol{Z}, \tag{5.9}$$

where $\boldsymbol{A}_m$ is a lower triangular matrix obtained by Cholesky decomposition of the covariance matrix, $\mathcal{K}_m = \boldsymbol{A}_m \boldsymbol{A}_m^T$, and $\boldsymbol{Z}$ is a vector of N standard normal variables $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

INITIALIZATION OF THE MIXTURE COMPONENTS. With the GP trajectory representation, selecting the initial mean is relatively straightforward since a constant-velocity straight line minimizes the joint accelerations and thus provides the notion of smoothness. However, with the MGP trajectory representation we have to select an initial mean for multiple GPs. We introduced the MGP representation in order to achieve better exploration and improve success rate in comparison to the state-of-the-art methods in finding feasible trajectories in complex environments. To accomplish this goal, we have to somewhat sacrifice smoothness since searching for trajectories that are far from the constant-velocity straight line in the configuration space requires larger joint accelerations. We propose an initialization technique of the MGP mixture components that tries to cover distant portions of the configuration space while retaining relatively small joint accelerations.

An initial mean μ_m is generated similarly to sampling a random trajectory with (5.9) but with a deterministic $\boldsymbol{Z}_m$

$$\boldsymbol{\mu}_m = \boldsymbol{\mu}_0 + \boldsymbol{A}_m \boldsymbol{Z}_m, \tag{5.10}$$

where $\boldsymbol{\mu}_0$ is a constant-velocity straight line in the configuration space. The elements of $\boldsymbol{Z}_m$ corresponding to the velocity of the m-th robot joint are set to either 1 or −1 during the first half of robot trajectory, and to the opposite value during the second half of trajectory (so that the total integral of injected noise is 0), while all other elements are set to 0. By exerting acceleration only on a single robot joint at a time, the initialized trajectories remain smooth, while still exhibiting significant deviation in the task space in different directions, allowing for better exploration of the environment. An example of the proposed initialization and its comparison to a single GP is shown in Fig. 5.1, both for an omnidirectional planar robot platform and a 7 DoF robot arm.

5.3.2 *Probabilistic inference with an MGP prior*

We formulate trajectory optimization as probabilistic inference as in Section 5.2, but instead of a GP prior we use the prior distribution given as an MGP with its mixing coefficients π_m, means $\boldsymbol{\mu}_m$, and covariances $\boldsymbol{\mathcal{K}}_m$. The conditional distribution $p(\boldsymbol{\theta}|\boldsymbol{z})$, given a particular value for $\boldsymbol{z}$, is the following mixture component

$$p(\boldsymbol{\theta}|z_m = 1) \propto \exp\{-\frac{1}{2}\left\|\boldsymbol{\theta} - \boldsymbol{\mu}_m\right\|^2_{\boldsymbol{\mathcal{K}}_m}\}, \tag{5.11}$$

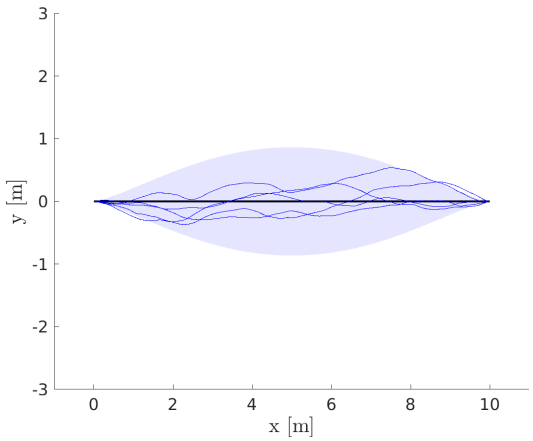
which corresponds to the GP prior in (3.4), similarly to the case of [59]. Then, the marginal distribution $p(\boldsymbol{\theta})$ that represents an MGP prior can then be written as

$$p(\boldsymbol{\theta}) \propto \sum_{m=1}^{M} \pi_m \exp\{-\frac{1}{2}\left\|\boldsymbol{\theta} - \boldsymbol{\mu}_m\right\|^2_{\boldsymbol{\mathcal{K}}_m}\}. \tag{5.12}$$

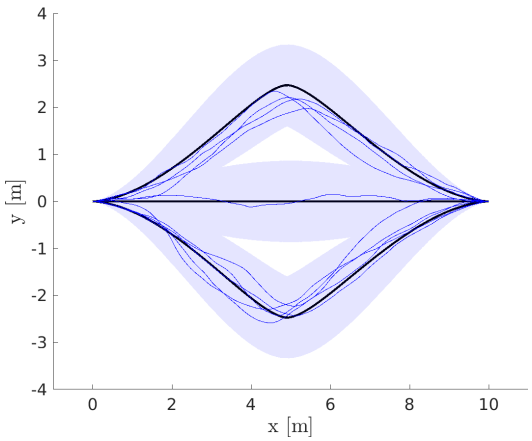
The optimal trajectory $\boldsymbol{\theta}^*$ is found by minimizing the negative logarithm of the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{e})$ given in (5.1)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left\{ -\log\left(p(\boldsymbol{\theta})p(\boldsymbol{e}|\boldsymbol{\theta})\right)\right\} \tag{5.13}$$

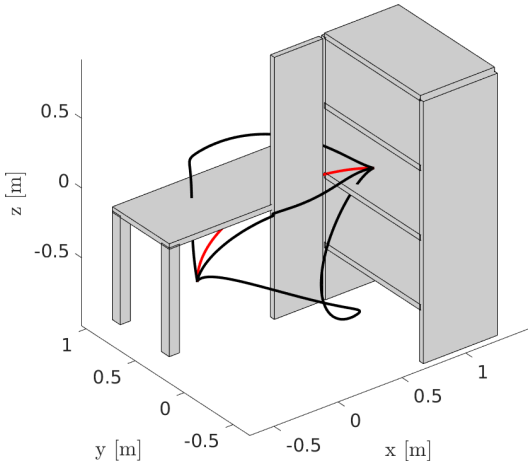
$$= \arg\min_{\boldsymbol{\theta}} \left\{ -\log \sum_{m=1}^{M} \pi_m \exp\{-\frac{1}{2}\left\|\boldsymbol{\theta} - \boldsymbol{\mu}_m\right\|^2_{\boldsymbol{\mathcal{K}}_m} - \frac{1}{2}\left\|\boldsymbol{h}(\boldsymbol{\theta}, \mathbf{e})\right\|^2_{\boldsymbol{\Sigma}}\}\right\}. \tag{5.14}$$



(a) 2D GP with a straight-line mean and its sample trajectories



(b) 2D MGP with three mixture components and its sample trajectories



(c) 7D MGP with four mixture components. Red trajectory is a straight line in the configuration space, while one of initial trajectories (black) is almost collision-free.

Figure 5.1: Comparison of GP and MGP initializations for a planar omnidirectional robot and a 7 DoF robot arm.

While the negative logarithm reduced the MAP problem to nonlinear least squares when a GP prior was used, in our case when the prior is an MGP, the negative logarithm does not simplify the problem due to the summation operator inside the logarithm. In the following we elaborate on how to solve this issue.

5.3.3 *Stochastic gradient estimation*

Suppose that we know the value of latent variable $\boldsymbol{z}$, i.e. which element $z_m = 1$, and that it remains constant. Then the value of m-th mixing coefficient becomes $p(z_m = 1) = \pi_m = 1$, while all other mixing coefficients are zero. In this case, the summation operator within the optimization problem in (5.14) disappears, and the problem becomes equivalent to (5.4)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \Big\{ \frac{1}{2} \big\| \boldsymbol{\theta} - \boldsymbol{\mu}_m \big\|^2_{\boldsymbol{\mathcal{K}}_m} + \frac{1}{2} \big\| \boldsymbol{h}(\boldsymbol{\theta}, \mathbf{e}) \big\|^2_{\boldsymbol{\Sigma}} \Big\}. \tag{5.15}$$

In order to allow optimizing trajectory costs that are non-differentiable, we employ derivative-free stochastic optimization, instead of the gradient descent optimization proposed in [59]. The introduced stochasticity could additionally enhance exploration and help alleviate the local minima problem present in gradient-based methods. Given that, first we convert the deterministic problem in (5.15) to a stochastic one by applying the expectation operator

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left\{ \mathbf{E}\left[\frac{1}{2} \left\| \boldsymbol{\theta} - \boldsymbol{\mu}_m \right\|^2_{\mathcal{K}_m} + \frac{1}{2} \| \boldsymbol{h}(\boldsymbol{\theta}, \mathbf{e}) \|^2_{\boldsymbol{\Sigma}} \right] \right\}. \tag{5.16}$$

Next we take the gradient with respect to $\boldsymbol{\theta}$ to minimize the pertaining expectation

$$\nabla_{\boldsymbol{\theta}} \mathbf{E}\left[(\boldsymbol{\theta} - \boldsymbol{\mu}_m)^T \mathcal{K}_m^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}_m) + \boldsymbol{h}^T(\boldsymbol{\theta}, \mathbf{e}) \boldsymbol{\Sigma}^{-1} \boldsymbol{h}(\boldsymbol{\theta}, \mathbf{e}) \right] = 0 \tag{5.17}$$

leading to

$$\mathbf{E}[\boldsymbol{\theta}] = \boldsymbol{\mu}_m - \mathcal{K}_m \underbrace{\mathbf{E}\left[\nabla_{\boldsymbol{\theta}} \left(\boldsymbol{h}^T(\boldsymbol{\theta}, \mathbf{e}) \boldsymbol{\Sigma}^{-1} \boldsymbol{h}(\boldsymbol{\theta}, \mathbf{e}) \right) \right]}_{\delta \hat{\boldsymbol{\theta}}_m, \text{ the gradient estimate}}. \tag{5.18}$$

To further simplify the gradient estimation problem, we assume that the cost $\boldsymbol{h}(\boldsymbol{\theta}, \mathbf{e})$ of each configuration $\boldsymbol{\theta}_i$ depends only on that configuration, meaning that future or past costs do not impact the current cost. This implies that the covariance matrix $\boldsymbol{\Sigma}$ is isotropic, i.e. $\boldsymbol{\Sigma} = \sigma \boldsymbol{I}$, leading to

$$\delta \hat{\boldsymbol{\theta}}_m = \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \left(\textstyle\sum_{i=1}^{N} \sigma h^2(\boldsymbol{\theta}_i) \right) \right], \tag{5.19}$$

where σ is a scalar parameter that can be utilized for cost weighting. In this chapter, we set $\sigma = 1$ for simplicity.

Existing trajectory optimization methods [47, 59] achieved computational efficiency by employing an iterative gradient descent update using an analytical gradient requiring a smooth differentiable optimization criteria. On the other hand, STOMP [71] proposed a gradient estimation update that circumvents the analytical requirement and can optimize non-differentiable costs. Inspired by STOMP and work in the path integral reinforcement learning [127], we propose an estimated gradient formulation as follows

$$\delta \hat{\boldsymbol{\theta}}_m = \sum_{k=1}^{K} z_m^k \boldsymbol{P}(\tilde{\boldsymbol{\theta}}^k)(\tilde{\boldsymbol{\theta}}^k - \boldsymbol{\mu}_m) \tag{5.20}$$

with probability metric

$$\boldsymbol{P}(\tilde{\boldsymbol{\theta}}^k) = \exp(-\frac{1}{\lambda} \boldsymbol{f}(\tilde{\boldsymbol{\theta}}^k)), \tag{5.21}$$

where $\tilde{\boldsymbol{\theta}}^k$ is a single sampled trajectory, z_m^k is the corresponding latent variable, K is the total number of sampled trajectories, and $\boldsymbol{f}(\tilde{\boldsymbol{\theta}}^k)$ corresponds to the state-dependent cost function, $\boldsymbol{f}(\tilde{\boldsymbol{\theta}}^k) = \sum_{i=1}^{N} \sigma h^2(\tilde{\boldsymbol{\theta}}_i^k)$. The parameter λ determines the sensitivity of the exponentiated cost. The variable z_m^k in (5.20) ensures that only samples from the m-th mixture component partake in gradient estimation for that component.

In other words, to estimate the gradient of the trajectory cost, first we draw K sample trajectories $\tilde{\boldsymbol{\theta}}^k$ and then evaluate the pertaining trajectory cost $\boldsymbol{f}(\tilde{\boldsymbol{\theta}}^k)$ for each sampled trajectory. Afterwards, the probability metric in (5.21), normalized to unity similarly as in [71],

is calculated for each trajectory cost and finally the gradient is computed following (5.20). Intuitively, trajectories with lower cost are desirable, as they are farther from the obstacles. If a trajectory cost of a certain sampled trajectory $\tilde{\boldsymbol{\theta}}^k$ is particularly small compared to other sampled trajectories, the probability $\boldsymbol{P}(\tilde{\boldsymbol{\theta}}^k)$ of that trajectory is very high. Then the trajectory cost gradient $\delta\hat{\boldsymbol{\theta}}_m$ points in the direction of that trajectory sample. The trajectory mean $\boldsymbol{\mu}_m$ updated with the pertaining gradient *moves* toward the part of trajectory space with smaller trajectory costs, i.e. with desirable properties. The proposed gradient estimation in (5.20) and update rule in (5.18) represent the generalization of STOMP for a continuous-time GP. Unlike the discrete-time trajectory representation utilized in STOMP, the continuous-time GP representation includes velocities which can be useful for optimizing costs that can explicitly reason about time [50]. It also supports the aforementioned efficient interpolation that can be exploited for querying the trajectory at any time of interest, resulting in dense collision checking and bypassing the need for post-processing.

5.3.4 *Updating the MGP weights during optimization*

The assumption that the variable $\boldsymbol{z}$ is constant and known allows us to carry out the same optimization technique when sampling from the whole MGP, and not just from a single mixture component. Unlike the previous section, where we had K trajectories from a single component, now we have K trajectories sampled from the whole mixture. Since the information about the trajectory origin, i.e. the corresponding latent variable z_m^k in known and kept, we can utilize trajectories sampled from the m-th mixture component to estimate the gradient and update the mean for that component. We do this for each of M mixture components, following the gradient estimation procedure (5.20). Essentially, the latent variable formulation enabled us to split the challenging optimization problem in (5.14) into M simpler ones that can be solved using the update rule (5.18).

However, we do not want to solve those M simpler problems independently. Since each mixture component covers a different part of trajectory space, some will have smaller cost, and therefore, higher chance of converging to a feasible solution trajectory. We want to explore around initializations that are closer to finding a feasible solution and neglect mixture components with higher trajectory cost. In other words, we also want to update the mixture weights between iterations to ensure that trajectories with smaller costs have higher mixture weights, so that computational time is not spent exploring around trajectories that have high costs, i.e. that are prone to collision. Therefore, we propose to update the mixture weights by utilizing a probabilistic metric similar to (5.21)

$$\pi_m = \exp\left(-\frac{1}{\lambda}\boldsymbol{f}(\boldsymbol{\mu}_m)\right). \tag{5.22}$$

Given that, after updating each of the M means $\boldsymbol{\mu}_m$, we evaluate their costs $\boldsymbol{f}(\boldsymbol{\mu}_m)$ and update their weights π_m. In the subsequent iterations the components with higher mixture weights get more trajectory samples and get more explored.

5.3.5 *MGPTO algorithm*

In this section we present the MGPTO method in the form of an algorithm, summarized in Algorithm 2, which uses results from previous subsections. As inputs, the algorithm requires

Algorithm 2 MGPTO

Input: Start and goal states $\boldsymbol{\theta}_0$, $\boldsymbol{\theta}_N$, a state-dependent cost function $f(\tilde{\boldsymbol{\theta}}^k(t))$
Precompute: Initial means $\boldsymbol{\mu}_m$ and covariances $\boldsymbol{\mathcal{K}_m}$

while convergence criterion not satisfied **do**
  for k = 1 . . . K **do**
    Draw a sample trajectory
    $\tilde{\boldsymbol{\theta}}^k(t) \sim \sum_{m=1}^{M} \pi_m \mathcal{GP}(\boldsymbol{\mu}(t), \mathcal{K}(t, t'))$
    Evaluate trajectory cost $f(\tilde{\boldsymbol{\theta}}^k(t))$
    Evaluate probability $\boldsymbol{P}(\tilde{\boldsymbol{\theta}}^k) = \exp(-\frac{1}{\lambda}\boldsymbol{f}(\tilde{\boldsymbol{\theta}}^k))$
  end for
  for m = 1 . . . M **do**
    Normalize probability $\boldsymbol{P}_m(\tilde{\boldsymbol{\theta}}^k) = \frac{z_m^k \boldsymbol{P}(\tilde{\boldsymbol{\theta}}^k)}{\sum_{k=1}^{K} z_m^k \boldsymbol{P}(\tilde{\boldsymbol{\theta}}^k)}$
    Estimate gradient
    $\delta\hat{\boldsymbol{\theta}}_m = \sum_{k=1}^{K} z_m^k \boldsymbol{P}_m(\tilde{\boldsymbol{\theta}}^k)(\tilde{\boldsymbol{\theta}}^k - \boldsymbol{\mu}_m)$
    Compute new mean $\boldsymbol{\mu}_m = \boldsymbol{\mu}_m + \boldsymbol{\mathcal{K}_m}\hat{\delta\boldsymbol{\theta}}_m$
    Evaluate cost of new mean $\boldsymbol{f}(\boldsymbol{\mu}_m)$
  end for
  Compute mixture weights $\pi_m = \frac{\exp(-\frac{1}{\lambda}\boldsymbol{f}(\boldsymbol{\mu}_m))}{\sum_{m=1}^{M}\exp(-\frac{1}{\lambda}\boldsymbol{f}(\boldsymbol{\mu}_m))}$
end while

a state-dependent cost function $f(\tilde{\boldsymbol{\theta}}^k)$ to be minimized as well as the start and goal states $\boldsymbol{\theta_0}$, $\boldsymbol{\theta}_N$ of the trajectory. A state-dependent cost function $f(\tilde{\boldsymbol{\theta}}^k)$ can be non-differentiable and pertain to any type of trajectory constraints or quality metrics. The mixture component means $\boldsymbol{\mu}_m$ are initialized using the technique proposed in Section 5.3.1. To initialize the mixture component covariances $\boldsymbol{\mathcal{K}}_m$, first the integral of the LTV-SDE in (3.5) is calculated at *support states* and the corresponding covariance matrix is written in the *lifted* form [63]. The underlying GP is then conditioned on the goal state to generate initializations of the covariance matrices $\boldsymbol{\mathcal{K}}_m$ that are suitable for motion planning [46, 112]. After initialization, the main loop of the algorithm is iterated until a given convergence criterion is satisfied. The main loop consists of two *for* loops. In the first *for* loop, the K trajectory samples $\tilde{\boldsymbol{\theta}}^k$ with the corresponding latent variables z_m^k are drawn with (5.9). For each sampled trajectory, its respective state-dependent cost is evaluated and the probability metric $\boldsymbol{P}(\tilde{\boldsymbol{\theta}}^k)$ is calculated following (5.21). In the second *for* loop, the gradient is estimated for each of the M mixture components using (5.20) and each of the M means are updated with

$$\boldsymbol{\mu}_m = \boldsymbol{\mu}_m + \boldsymbol{\mathcal{K}_m}\hat{\delta\boldsymbol{\theta}}_m. \tag{5.23}$$

After updating, the state-dependent cost of each new mean $\boldsymbol{f}(\boldsymbol{\mu}_m)$ is calculated, which is used at the end to compute the new mixture weights following (5.22). Note that we also normalize (5.22) to unity for correct probability representation.

5.4 IMPLEMENTATION DETAILS OF THE PROPOSED METHOD

In this section we discuss the implementation details necessary for better comprehension and full reproducibility of the proposed MGPTO method. Note that encoding the collision avoidance criterion, we rely on the obstacle cost function defined in (3.22), with robot's body represented as a set of spheres as explained in Section 3.2.1 and an SDF environment representation presented in Section 3.2.2.

5.4.1 *Convergence criteria*

In Algorithm 2, the optimization runs until a convergence criterion is satisfied. There are several different convergence criteria that we implement for different purposes. For example, when searching for a single collision-free trajectory, we terminate the optimization when one of the means $\boldsymbol{\mu}_m$ of mixture components becomes collision free, i.e. its collisions cost becomes 0, and that mean is outputted as the solution trajectory. Alternatively, our approach also allows to search for multiple different solution trajectories, i.e., when a collision-free mean $\boldsymbol{\mu}_m$ is found, the pertaining m-th mixture component is pruned from the mixture and the mixture weights π_m for the remaining components are recalculated. The proposed convergence criterion can thus be utilized to find multiple modes of the cost function corresponding to multiple solution trajectories, which has not been extensively studied in the literature. Multiple different solutions allow selecting a single solution based on how the trajectory may appear, which homotopy class it should belong or according to secondary criteria, e.g. the distance from singular configurations or energy consumption. Selecting one of the solutions obtained by MGPTO circumvents the need to replan with different initialization or cost function which might be the case with methods that produce a single solution. Furthermore, if changes in the environment that block a certain path occur during planning time, having alternative solutions that avoid the blocked pathway bypasses the need for replanning. Multiple solutions may also be of use in task-level motion planning [78]. In any case, we can terminate the optimization if too much computational effort was spent by setting a t_{max} parameter for maximum allowed processing time. When optimizing secondary costs (e.g. motor torques), the algorithm stops when the trajectory cost converges, i.e. when it does not decrease for several iterations.

5.4.2 *Torque optimization*

Given a robot dynamics model, the feed-forward torque, needed at each joint to track the desired trajectory, can be computed using inverse dynamics algorithms [128]. The motor torques at any time instant t_i are function of the robot joint states and the pertaining derivatives

$$\boldsymbol{\tau}(t_i) = f(\boldsymbol{\theta}_i, \dot{\boldsymbol{\theta}}_i, \ddot{\boldsymbol{\theta}}_i). \tag{5.24}$$

While the joint states and joint velocities at time instant t_i are a part of each sampled trajectory due to the underlying constant-velocity model, joint accelerations are calculated by forward finite differentiation.

We construct the cost function that is utilized to minimize the robot motor torques

similarly as in [71], by adding the magnitude of torque for each of the J robot joints

$$h(\boldsymbol{\theta}_i) = \sum_{j=1}^{J} \left| \boldsymbol{\tau}_j(t_i) \right| . \tag{5.25}$$

Obtaining torque derivations can be challenging and the absolute value operator makes the cost function in (5.25) non-differentiable at 0. This is an example of a state-dependent cost that the proposed stochastic optimization method can optimize, unlike the gradient-based methods. Even if we define a differentiable cost function, analytical derivatives of the inverse dynamics function (5.24) are often unavailable in practice, incurring additional computational burden for gradient-based methods.

5.4.3 *Geometry-aware singularity avoidance*

The geometry-aware singularity index presented in Section 3.3 avoids the failure modes inherent to manipulability and dexterity when detecting kinematic singularities. However, it is yet to be exploited in the context of motion planning. Herein, we derive a cost function that features the geometry-aware singularity index, which provides a measure of proximity to singular configurations.

To integrate singularity avoidance with trajectory optimization, we propose a cost function that takes into account the value of the geometry-aware singularity index. Combining Eq. (3.31) and Eq. (3.32) and summing over every support state $\boldsymbol{\theta}_i$, we obtain

$$h(\boldsymbol{\theta}_i) = \sum_{i=0}^{N} \left\| \log\left(k^{-\frac{1}{4}} M(\boldsymbol{q}_i) \right) \right\|_F^2 . \tag{5.26}$$

Note that only configuration part $\boldsymbol{q}_i$ of the state $\boldsymbol{\theta}_i$ is used for its calculation, while information about velocity $\dot{\boldsymbol{q}}_i$ is not needed. The cost function in Eq. (5.26) should guide the trajectory optimization away from regions of configuration space containing singularities without being prone to failure modes of manipulability and dexterity indices.

5.4.4 *Joint limits*

MGPTO method deals with joint limits by clamping the sampled trajectories $\tilde{\boldsymbol{\theta}}^k$ at the limit values, similarly as in [71]. The updated means $\boldsymbol{\mu}_m$ at each iteration respect the joint limits, since the sampled trajectories $\tilde{\boldsymbol{\theta}}^k$ stay within the limits and the stochastic gradient update is essentially a convex combination of sampled trajectories. Moreover, since the estimated gradient $\delta\hat{\boldsymbol{\theta}}_m$ is smoothed through the covariance matrix $\boldsymbol{\mathcal{K}}_m$ with the underlying constant-velocity model, the updated means $\boldsymbol{\mu}_m$ smoothly touch the joint limit, thus avoiding jerky motion. Given that, MGPTO handles the joint limits during optimization, unlike GPMP2 and CHOMP that handle joint violations as a post-processing step by smoothly projecting joint violations on the output trajectory, which can sometimes lead to the post-processed trajectories that are in collision. Moreover, another advantage of the MGP representation is that it includes velocities as part of the state-space, unlike the discrete representation in STOMP, meaning that MGPTO can also enforce joint angular velocity limits in the same manner, which STOMP is currently unable to do.

5.5 EXPERIMENTAL RESULTS

In this section we present the results of experimental validation of the proposed method and its comparison to the state-of-the-art trajectory optimization techniques GPMP2 [59], STOMP [71] and the method [112] presented in Chapter 5. We conducted six experiments in different motion planning scenarios:

- *box obstacle benchmark*,
- *robot arm in a complex environment benchmark*,
- *torque optimization benchmark*,
- *singularity avoidance benchmark*,
- *mobile manipulator scenario*,
- *dual-arm robot real-world experiment*.

The aim of the *box obstacle benchmark* was to show the benefits of the proposed mixture representation, which allows for determining multiple modes of the cost function unlike the state-of-the-art methods relying on a single straight-line initialization. On this experiment we also conducted extensive analysis of MGPTO hyperparameters to determine their optimal values for achieving the best performance. The aim of the *robot arm in a complex environment benchmark* was to attest the benefits of the MGP prior in aiding better exploration and avoiding local minima to find feasible solutions to the planning problem. The *torque optimization benchmark* aimed to show the ability of MGPTO to optimize a cost function consisting of multiple cost functionals which may be non-differentiable. State-of-the-art gradient-based methods (TrajOpt, CHOMP, GPMP2) are not able to optimize such costs. The *singularity avoidance benchmark* aimed to additionally corroborate the ability of MGPTO to optimize different cost functionals, specifically targeting kinematic singularity avoidance during the robot arm motion. The goal of the *mobile manipulator scenario* experiment with a 10 DoF mobile manipulator was to showcase the ability of MGPTO to successfully find solution trajectories for high DoF robots in cluttered environments. And finally, the *dual-arm robot real-world experiment* aimed to verify the effectiveness of MGPTO for high-dimensional motion planning in a real-world environment.

For GPMP2, we used its open-source C++ library [59, 120] and the respective MATLAB toolbox based on the GTSAM library [121], while we opted for our own implementation of STOMP and the method in [112]. All experiments were performed on a single core of a 2.8-GHz Intel Core i7-7700HQ processor with 16 GB of RAM. In our evaluation, every mixture component was initialized with the same covariance matrix $\mathcal{K}_m$, instead of having M different ones. We also used the aforementioned efficient GP interpolation for dense collision checking, similarly as in [59]. Since the *support states* of sampled trajectories are equidistant in time and the covariance matrix $\mathcal{K}_m$ is constant and time-invariant for every mixture component, we precomputed the employed interpolation matrices [59], instead of computing them every time interpolation was needed. This provided a significant boost to the computational efficiency of MGPTO.

Table 5.1: Performance of MGPTO with various combination of hyperparameters and comparison with trajectory optimization methods GPMP2, STOMP and the method in [112] in the box obstacle scenario.

	MGPTO												GPMP2	STOMP	Method in [112]
M	3	5	10	15	15	15	15	15	15	15	15	15			
K	50	50	50	50	30	70	100	200	50	50	50	50			
Q_c	1	1	1	1	1	1	1	1	0.1	2	5	10			
Success rate (%)	42	90	90	**100**	**100**	**100**	97	80	87	**100**	**100**	98	**100**	47	**100**
Time to first solution (ms)	433	452	452	177	203	312	339	720	290	238	304	330	**94**	469	252
Average number of solutions	0.6	1.52	2.73	**6.33**	5.9	4.4	3.7	1.83	2.96	5.61	4.82	4.55	1	1	1

5.5.1 *Box obstacle benchmark*

The first benchmark consisted of a simple environment with a single floating box obstacle, as shown in Figure 5.2a. We conducted 100 unique planning experiments, all with different start and goal states with start points being on the left-hand side of the box and end points being on its right-hand side. To make this experiment non-trivial, for every planning problem we ensured that the robot arm is relatively close to the box obstacle at start and end points and that the initial constant-velocity straight line passes through the box obstacle. While this problem is relatively easy to solve for state-of-the-art methods, benchmarking the algorithm in this environment has several advantages. First, it is easily reproducible due to the simplicity of the environment and thus allows for a fair comparison. A box in the robot's workspace can be seen as a generalization of any obstacle. For this reason we conducted a comprehensive hyperparameter analysis of MGPTO on this benchmark as its simplicity allows to gain intuitive insight about the impact of various hyperparameters. Furthermore, there are multiple solutions to the problem that are obvious to a human – a robot arm can move over, under or behind the box. The proposed setup can thus be utilized to demonstrate the ability of MGPTO to find multiple modes of the cost function corresponding to multiple solution trajectories.

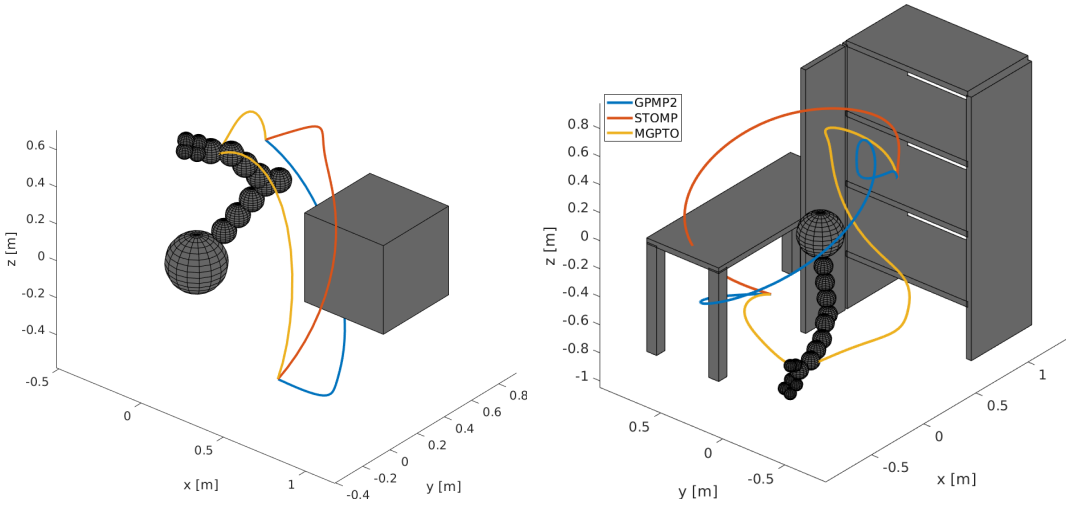
In this experiment we measured the success rate, average algorithm execution time, and the average number of different solution trajectories for MGPTO and compare it to GPMP2, STOMP and the method in [112]. The average execution time for MGPTO corresponds to the time to find the first feasible solution. Due to the simplicity of the experiment, we set the maximum execution time for every method to $t_{max} = 1\,\text{s}$ after which the optimization is terminated. We evaluated the performance of MGPTO with regards to its three hyperparameters: (i) M, the number of mixture components, required for mixture initialization in (5.5) and further mixture updates in (5.22), (ii) K, the number of trajectories sampled at each iteration, required for gradient estimation in (5.20), (iii) Q_c, the power spectral density matrix parameter of the underlying white noise process which determines how noisy the sampled trajectories are, required in (3.5). We tested twelve different parameter sets, evaluating the impact of each parameter with four different values, *ceteris paribus*. For MGPTO, GPMP2 and the method in [112], we set the total trajectory time, i.e. the timespan in which the robot moves from start to goal state, to $t_{total} = 10\,\text{s}$ and we parameterized trajectories with 10 *support states* with 10 interpolation states in-between for which the trajectory cost was evaluated, resulting in total of 101 states. Using less support states with more interpolated states in-between leads to the optimal performance of GPMP2 [46] and this also applies to MGPTO since we utilize a similar underlying GP framework.

We parameterized the STOMP trajectory with $N = 100$ discrete states. For GPMP2, the parameters $\epsilon = 0.2$ and $\Sigma = 0.2$ were chosen for which we empirically determined that they lead to the best results in this experiment. For STOMP, the parameter that determines the number of noisy trajectories generated at each iteration was set to $K = 10$, which was demonstrated to achieve good performance [71]. For the method in [112], we empirically determined its parameter $K = 200$ that led to the best results in this experiment. In all of our experiments, we set the parameter λ that determines the sensitivity of the cost in (5.21) as $\lambda = 0.1$, which was also chosen in [71] as the optimal value.

The results of this experiment are shown in Table 5.1. MGPTO was able to solve every planning problem with multiple combinations of parameters. Out of four tested values for parameter determining the number of mixture components, $M = 15$ demonstrated the best performance. More different mixture initializations lead to faster exploration of the environment and to a larger number of different solutions found. Note that the maximum number of different mixture initializations in this case was exactly 15 since our initialization technique proposed in Section 5.3.1 exerts acceleration only on a single robot joint at a time to keep the trajectory smooth. Since there are 7 robot joints, there is a maximum of 15 initializations; one constant-velocity straight-line and two initializations for every robot joint exerting either positive or negative acceleration during the first half of robot trajectory. The optimal value of parameter K was found in the range from 30 to 70. If there are multiple mixture initializations, then the number of trajectories sampled at each iteration K smaller than 30 does not allow for exploration around every initialization. If K is too large then much of the computation time is spent on evaluating the cost for large number of trajectories which slows down the convergence. The optimal value of parameter Q_c was found in the range from 1 to 5. For small values of Q_c, MGPTO does not explore much around initial trajectories and sometimes fails to find collision-free trajectories. For values of Q_c larger than 5 there is too much noise in the sampled trajectories which sometimes do not even resemble the initial trajectory around which they were sampled. In such cases MGPTO becomes reliant on *luck* which means that it sometimes fails to find collision-free trajectories and that two different runs of the method result in potentially vastly different solutions. MGPTO greatly outperformed STOMP which often failed to find a collision-free solution. STOMP was particularly troubled when the start or end point was initially close to the box obstacle. GPMP2 solved every planning problem and it was almost twice as fast as MGPTO, while the method in [112] demonstrated similar performance to MGPTO in this benchmark. However, the advantage of MGPTO stems from the fact that the mixture representation allowed finding multiple solutions, while GPMP2 and the method in [112] went above the box obstacle in most cases. Figure 5.2a presents an example where MGPTO finds three qualitatively different solutions.

5.5.2 *Robot arm in a complex environment benchmark*

The robot arm planning benchmark consisted of a simulated WAM robotic arm in an environment featuring a table and a drawer cabinet, as shown in Figure 5.2b. We conducted 100 unique planning experiments, all with different start and goal states with start states being near the table and end states being inside the cabinet. This is a typical scenario for



(a) An example of three different solution trajectories in the *box obstacle benchmark*

(b) An example where MGPTO finds a collision-free trajectory while GPMP2 and STOMP fail in the *robot arm in a complex scenario benchmark*

Figure 5.2: A simulated WAM arm in two scenarios: (a) an environment featuring a single box obstacle (first benchmark), (b) an environment featuring a table and a drawer (second benchmark). Plotted lines depict the end effector trajectories.

benchmarking trajectory optimization methods, featuring a realistic environment with challenging local minima.

In this benchmark we compared MGPTO with GPMP2, STOMP, the method [112] presented in Chapter 5 and GPMP2 with random restarts (GPMP2-RR) that is a commonly used approach to alleviate the local minima problems in gradient-based trajectory optimization [47]. Following our hyperparameter analysis in Section 5.5.1, for MGPTO we chose the number of mixture components $M = 15$ and the number of sampled trajectories in each iteration $K = 50$. The experimental setup in this benchmark is more complex than the box scenario so larger Q_c is warranted for better exploration and being less prone to the local minima. We empirically determined that $Q_c = 2$ leads to the best performance. As in our previous experiments, we set the total trajectory time $t_{\text{total}} = 10$ s. The trajectories for MGPTO, GPMP2 and the method in [112] were parameterized with $N = 10$ equidistant *support states* and 10 interpolation steps in-between. For GPMP2, the parameters $\epsilon = 0.2$ and $\Sigma = 0.2$ were chosen for which we empirically determined that they lead to the best results on this experiment. In GPMP2-RR, the optimizer is first initialized with a straight-line and then, on failure, reinitialized with a random trajectory. We allowed for a maximum of 20 restarts. For the method [112] presented in Chapter 5, we empirically determined its parameter $K = 400$ that led to the best results in this experiment. For STOMP, the parameter K was set to $K = 10$, which was demonstrated to achieve good performance [71]. Due to stochastic nature of results, we repeated every planning experiment with MGPTO, GPMP2-RR, the method in [112] and STOMP 10 times to correctly assess the success rate and computation time. We set the maximum execution time for every method to $t_{max} = 3$ s after which the optimization is terminated.

The results of the experiment are shown in Table 5.2. MGPTO outperformed GPMP2

Table 5.2: Comparison of MGPTO with GPMP2, STOMP and the method in [112] in robot arm planning benchmark

	MGPTO	GPMP2	GPMP2-RR	STOMP	Method in [112]
Success rate (%)	98	51	86	35	**100**
Execution time (ms)	792	**82**	460	1844	1471

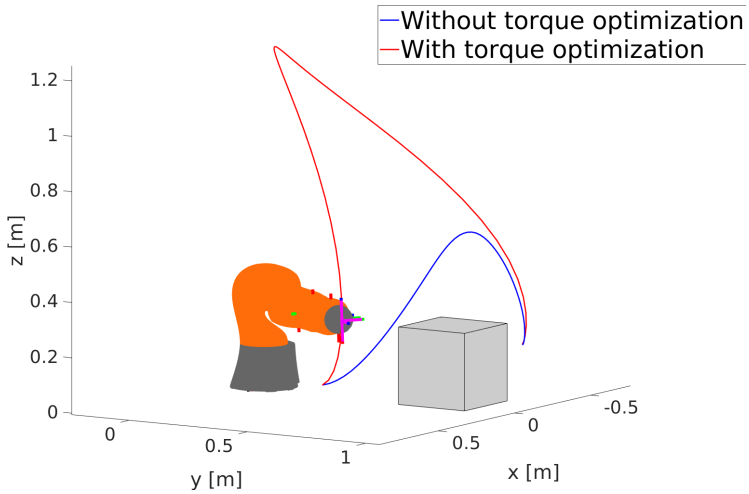
and STOMP regarding the success rate. STOMP achieved both the worst success rate and execution time due to its inability to explore. While the baseline GPMP2 often fails, the stochasticity introduced by random restarts helps in achieving higher success rates. GPMP2 is an order of magnitude faster than MGPTO and STOMP. This is expected since it is a classical gradient-based trajectory optimization method. The method in [112] is well-suited for this type of problem and was able to solve every problem instance but required significantly more computation time than MGPTO. Note that MGPTO was actually able to solve every problem in the dataset on separate occasions, but it did not solve every problem every time and thus the reported success rate is not 100%. However, the sheer ability to solve every problem on this dataset is an indication of benefits of the MGP prior in aiding better exploration of the environment. It also implies that, on failure, MGPTO can be executed again to retry finding a solution trajectory. An example of a case where our method finds a solution, while GPMP2 and STOMP fail is depicted in Figure 5.2b. The trajectory obtained with method in [112] was qualitatively similar to the solution found by MGPTO and thus we omit it for clarity.

5.5.3 *Torque optimization benchmark*

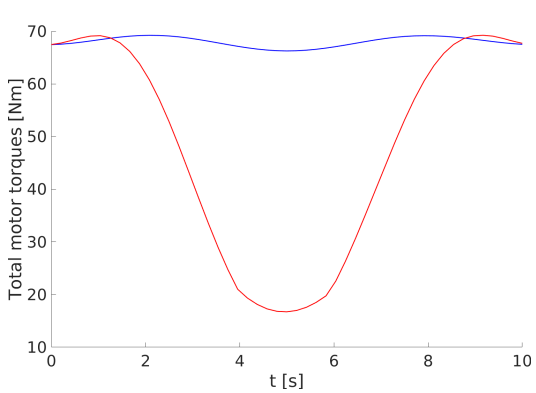
The torque optimization benchmark featured a simulated 7 DoF Kuka LBR iiwa robotic arm in an environment featuring a single box (Figure 5.3a), similar to the setup in Section 5.5.1. We conducted 100 unique planning experiments with different start and goal states where the robot arm starts from the left side of the box and finishes on the right side. Besides collision avoidance, the optimized trajectory cost $\boldsymbol{f}(\tilde{\boldsymbol{\theta}}^k)$, required in (5.21), included the criteria for torque minimization, described in (5.25).

In this experiment we tested MGPTO with and without torque optimization cost in order to demonstrate its effectiveness in minimizing secondary costs that are non-differentiable. We compared the results with STOMP since it is, to the best of authors' knowledge, the only state-of-the-art trajectory optimization method that demonstrated the ability to minimize motor torques. As in our previous experiments, we set the total trajectory time $t_{\text{total}} = 10$ s and we parameterized the trajectory with $N = 10$ equidistant *support states* with 10 interpolation steps in-between. Guided by our hyperparameter analysis in Section 5.5.1, we chose the number of mixture components $M = 15$, the number of sampled trajectories in each iteration $K = 50$ and the covariance matrix parameter of the white noise $Q_c = 1$.

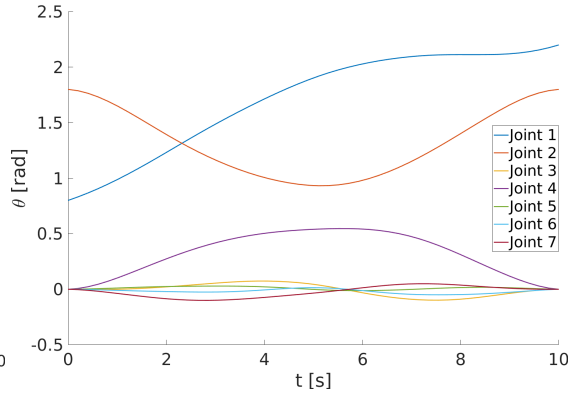
The average sum of absolute motor torques and the average execution times of MGPTO and STOMP in this experiment are shown in Table 5.3. The MGPTO with torque optimization improved the average sum of absolute motor torques during the trajectory around 30% when compared to MGPTO without torque optimization, and around 4% when compared to STOMP. Furthermore, the runtime of MGPTO compared to STOMP was significantly



(a) An example of obtained trajectories



(b) Sum of absolute motor torques through time



(c) Joint angles through time

Figure 5.3: Comparison of MGPTO with and without torque optimization in an environment featuring a single box obstacle (*torque optimization benchmark*).

shorter which stems from the fact that some of the mixture components in MGPTO initially cover parts of the configuration space with lower torques, while STOMP needs more time to explore. MGPTO thus converges faster, attesting the benefits of using mixtures instead of a single straight line initialization. An example of the obtained trajectories in simulation is shown in Figure 5.3a, while the corresponding sum of absolute robot motor torques is shown in Figure 5.3b. When the torque cost was optimized, the resulting trajectory moved the robot arm upwards resulting in configurations that required lower gravity compensation torques. Since the arm movements are relatively slow, most of the motor torques are used for gravity compensation and thus torques near the start and end states remain the same with and without optimization. We omit the trajectory obtained with STOMP from the illustration since it was qualitatively very similar to the trajectory obtained with MGPTO. Joint angles through time for the torque-optimized trajectory, depicted in Figure 5.3c, demonstrate that our trajectories retain smoothness even when multiple different optimization criteria are included in the cost function.

We also conducted a real-world experiment featuring the 7 DoF right arm of the Rethink Robotics Baxter robot. The resulting trajectory and the corresponding sum of absolute motor torques through time are shown in Figure 5.4 and the accompanying video. The results of

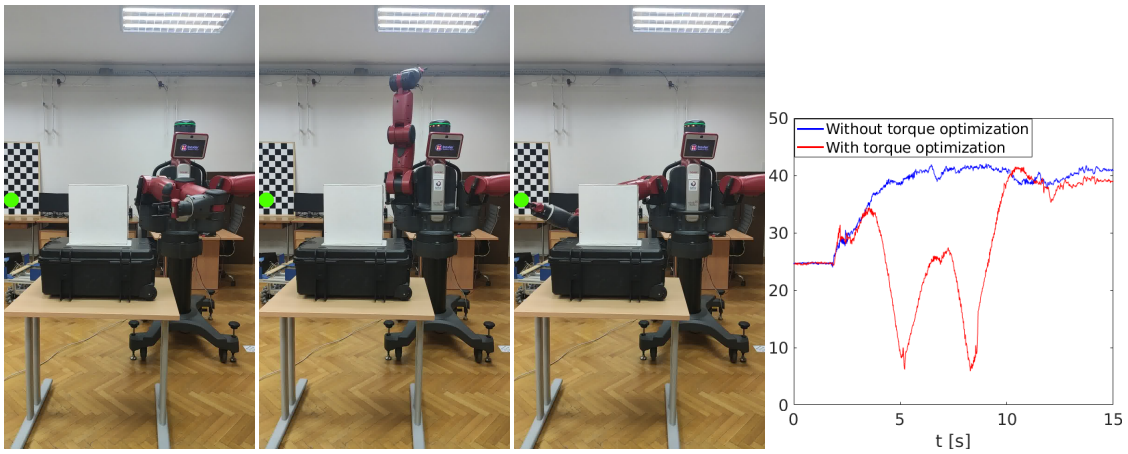


Figure 5.4: Baxter's right arm configurations during execution of a trajectory obtained with MGPTO and the sum of absolute motor torques. The green dot depicts end-effector's goal pose.

Table 5.3: Comparison of MGPTO with STOMP in torque optimization benchmark

	MGPTO	STOMP	No torque cost
Avg. sum of torques (Nm)	**42.34**	44.02	60.24
Execution time (s)	5.66	11.98	**0.44**

the experiment were qualitatively similar to those obtained in simulation, thus verifying MGPTO's performance in real-world scenarios.

5.5.4 *Singularity avoidance experiments*

We evaluated MGPTO's ability to avoid kinematic singularities in simulations on two different motion planning scenarios. First scenario featured a reaching task in an empty environment, where the aim was to reach a desired goal state while successfully avoiding singularities. Therefore, we provide a quantitative performance analysis of MGPTO with singularity avoidance cost function presented in (5.26) on the reaching task scenario and comparison with two state-of-the-art approaches to singularity avoidance, namely [69] and [107]. Second scenario featured a singularity and collision avoidance task, where the aim was to reach the desired goal state without colliding with the obstacle while avoiding singularities, to yet again showcase the ability of multiple-criteria optimization. To this end we provide a qualitative example of the second scenario that demonstrated the ability of the proposed method to find a collision free trajectory while staying far from singularities at every trajectory state.

The reaching task benchmark was performed using three common robot arms: the 6 DoF Universal Robots UR10, the 7 DoF Kinova Jaco 2 with spherical joints and the 7 DoF Kuka LBR IIWA 14. For each arm, the evaluation consisted of 100 different tasks with start and goal configurations randomly sampled from the robot's configuration space. We solved each task with MGPTO with geometry-aware singularity avoidance cost, the trajectory optimization method with manipulability optimization proposed in [69] and the singularity-robust kinematic control method proposed in [107].

We compared performance of these methods on three common singularity indices

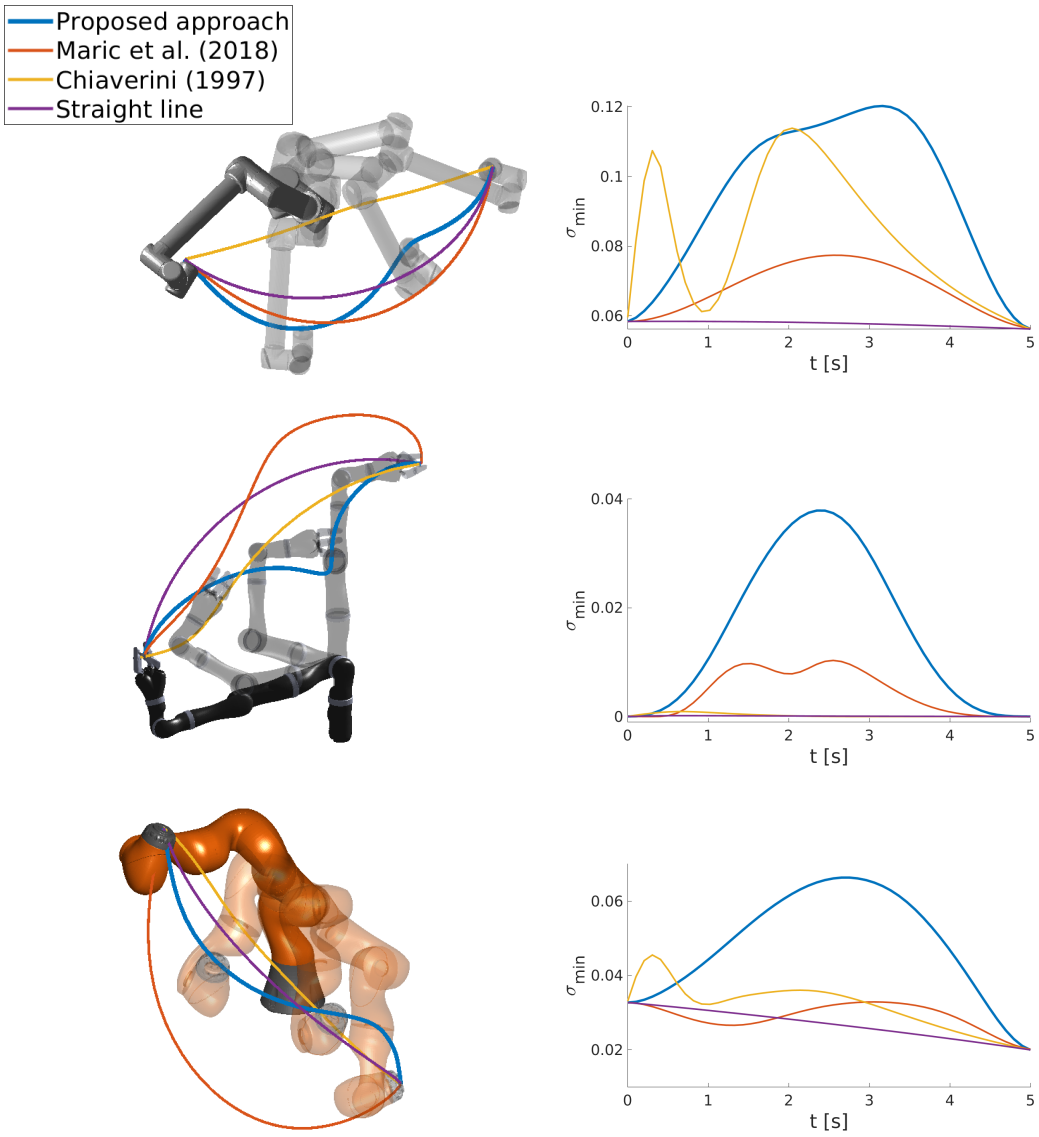


Figure 5.5: Example trajectories obtained by each singularity avoidance method on the reaching task. Universal Robots UR10 is shown at the top, Kinova Jaco 2 is depicted in the middle and Kuka LBR IIWA 14 is shown in the bottom area of the figure. The figures on the left show the end effector trajectories in robot's task space, while figures on the right show the corresponding minimum eigenvalue of the manipulability ellipsoid during trajectory execution.

which served as performance metrics: the minimum eigenvalue σ_{min} of the manipulability ellipsoid $\mathbf{M}(\mathbf{q})$ defined in Eq. (3.28), the manipulability index m defined in Eq. (3.29) and the dexterity index κ defined in Eq. (3.30). For every trajectory, we evaluated the performance metric at every discrete time instant and calculated the average value during the whole trajectory. Additionally, we provide performance analysis for a simple straight-line trajectory in configuration space as the baseline. We also compare execution times of evaluated methods, except for the baseline, as the straight-line trajectory can be straightforwardly calculated by linear interpolation in sub-millisecond time. Examples of trajectories obtained by all the methods for each of the employed robot arms are depicted in Fig. 5.5. For these

Table 5.4: Comparison of the proposed approach and several other methods on the reaching task benchmark for three different robot arms. We compare the methods with three different measures, namely the average minimal singular value, the average manipulability and the average dexterity.

	UR-10				Kinova Jaco 2				Kuka IIWA 7			
	σ_{min}	m	κ	Time	σ_{min}	m	κ	Time	σ_{min}	m	κ	Time
Proposed approach	**0.044**	**0.130**	**2.339**	114	**0.027**	**0.111**	**2.364**	163	0.025	**0.076**	**2.404**	126
Maric et al. (2019), [69]	0.034	0.111	2.518	**66**	0.025	0.099	2.380	**72**	0.017	0.057	2.621	**69**
Chiaverini (1997), [107]	0.030	0.072	2.801	129	0.023	0.073	2.440	182	**0.026**	0.061	2.421	138
Straight line	0.031	0.096	2.716	/	0.021	0.090	2.583	/	0.013	0.049	2.851	/

example trajectories, Fig. 5.5 also shows the corresponding minimum eigenvalue σ_{min} of the manipulability ellipsoid $\mathbf{M}(\mathbf{q})$ during trajectory execution. Results of the performance evaluation are shown in Table 5.4. Note that for minimum eigenvalue σ_{min} and manipulability index m higher values imply better singularity avoidance, while for dexterity index κ lower values are better.

As expected, each of the evaluated methods compared favorably to the baseline in every performance metric. The trajectory optimization method with manipulability optimization proposed in [69] attained the lowest average execution time for each of the utilized robot arms. This was expected due to its underlying gradient-based optimization. However, it is clear that the proposed method consistently achieved the best overall performance on the conducted reaching tasks. The coupling of MGPTO with a Riemannian-based geometry-aware singularity index proved to work notably well for singularity avoidance.

The singularity and collision avoidance task was performed using the 7 DoF Kuka LBR IIWA 14 robot arm in an environment featuring a table. We picked a start state under the table and the goal state above the table in order to require collision avoidance to successfully solve the task. We ran trajectory optimization twice: first, including both collision avoidance and our geometry-aware singularity avoidance in the cost function, and second, only considering collision avoidance. Examples of trajectories obtained by these two runs are depicted in Fig. 5.6, as well as the corresponding minimum eigenvalue σ_{min} of the manipulability ellipsoid $\mathbf{M}(\mathbf{q})$, manipulability index m and dexterity index κ during trajectory execution.

It can be seen that the inclusion of the singularity avoidance cost led to the robot arm to stay far away from singularities at every trajectory state, while retaining the ability of the motion planning method to find a collision-free trajectory. Better values of manipulability and dexterity indices also suggest that the robot's agility would be ensured in case of unpredictable changes.

5.5.5 *Mobile manipulator experiment*

We conducted the mobile manipulator trajectory planning experiment to demonstrate the ability of MGPTO to find collision-free trajectories for high DoF robots in complex environments. The simulated mobile manipulator consisted of a simulated WAM robotic arm mounted on top of an omnidirectional platform in an environment featuring two tables, a drawer cabinet and a generic static obstacle (see Figure 5.7). The system has 10 DoF corresponding to mobile platform's 3 DoF pose and robot arm's 7 joint angles. We planned

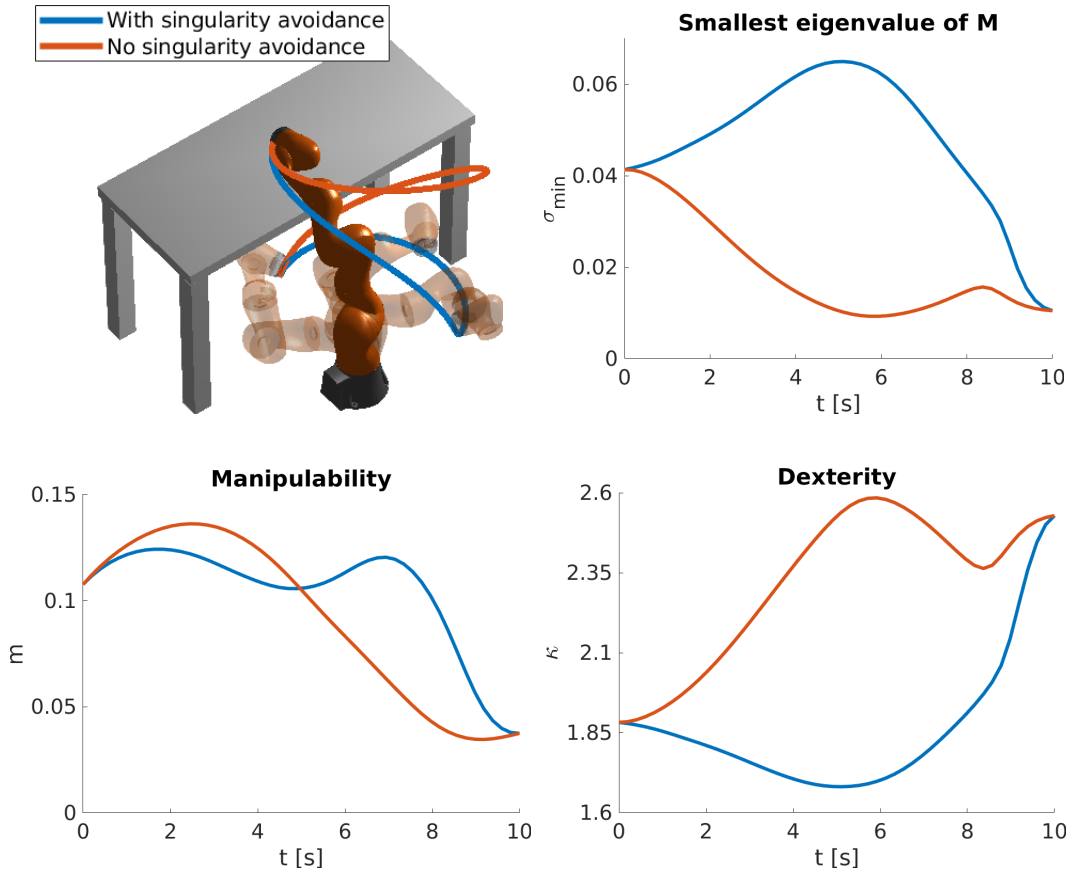
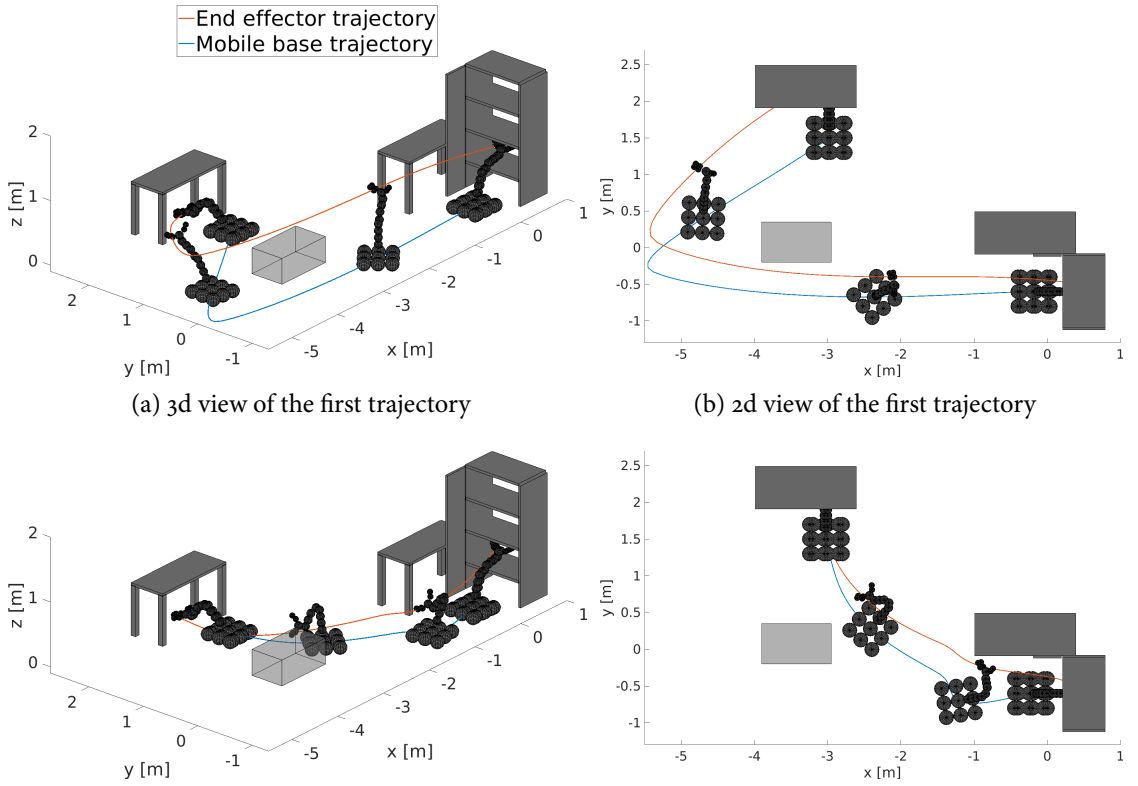


Figure 5.6: Trajectories obtained on the collision avoidance task. The top left figure shows the end effector trajectories in the robot's task space, while other figures correspond to the minimum eigenvalue of the manipulability ellipsoid, manipulability index and dexterity index during trajectory execution for trajectories obtained by the proposed stochastic trajectory optimization method with and without singularity avoidance optimization.

the trajectory of the whole system, simultaneously planning the motion of the platform and the robotic arm.

We chose the number of mixture components $M = 5$. This choice of M leads to initial mixture component trajectories differing in trajectories of the mobile platform, while the initial trajectories of the robot arm are locally identical. Using more mixture components, which would change initial trajectories of the attached robot arm, would not lead to noticeably better exploration since the movement of the mobile platform has more impact on the global pose of the whole mobile manipulator than the movement of the attached robotic arm. The number of sampled trajectories in each iteration was set to $K = 30$, while the covariance matrix parameter of the white noise was set to $Q_c = 1$. For both parameters we empirically determined that they lead to satisfactory performance with regards to computation time. As in our previous experiments, we set the total trajectory time $t_{\text{total}} = 10$ s. The trajectory was parameterized with $N = 20$ equidistant *support states* and 10 interpolation steps in-between. The maximum execution time was set to $t_{max} = 5$ s. To assess the required computation time, we repeated this experiment 30 times. Every take resulted in successfully finding at least one collision-free trajectory and the average algorithm time to first solution was 841 ms, while the worst case computation time was 988 ms. In most cases, MGPTO was able to find



(a) 3d view of the first trajectory

(b) 2d view of the first trajectory

(c) 3d view of the second trajectory

(d) 2d view of the second trajectory

Figure 5.7: An example of two trajectories obtained with MGPTO on a simulated mobile manipulator in a static environment.

two distinct trajectories belonging to different homotopy classes. The resulting trajectories are shown in Figure 5.7. The trajectory in Figures 5.7a, 5.7b was found every time, while the trajectory in Figures 5.7c, 5.7d was found in 23 out of 30 experiment runs. The average computation time to find the second solution trajectory was 1801 ms.

5.5.6 *Dual-arm robot real-world experiment*

We conducted the dual-arm robot real-world experiment to demonstrate the ability of MGPTO to find collision-free trajectories for high DoF robots in real environments. We planned the motion of the Rethink Robotics Baxter robot that features two 7 DoF robotic arms giving it a total of 14 DoF. Our laboratory environment consisted of a table and a drawer, resembling the simulated environment used in Section 5.5.2. We planned the trajectory of the whole system, simultaneously planning the motion of both arms. The left arm was tasked with moving from the top shelf of the drawer to the left side of the robot, while the right arm simultaneously moved from the table towards the middle shelf of the drawer.

As in our previous experiments, we set the total trajectory time $t_{\text{total}} = 10\,\text{s}$. The trajectories were parameterized with $N = 10$ equidistant *support states* and 10 interpolation steps in-between. The computation time to find the solution trajectory was 2404 ms with parameters $M = 15$, $K = 50$ and $Q_c = 1$. The results of the experiment are shown in Figure 5.8 and the accompanying video.



Figure 5.8: Baxter's dual-arm configurations during execution of a trajectory obtained with MGPTO. The left arm was tasked with moving from the top shelf of the drawer to the left side of the robot (green dot), while the right arm simultaneously moved from the table towards the middle shelf of the drawer (yellow dot).

5.5.7 *Discussion*

In the *box obstacle benchmark*, we demonstrated the ability of MGPTO to find multiple solutions to the planning problem. This feature is unique to MGPTO as none of the state-of-the-art trajectory optimization methods are able to do so. The *robot arm in a complex environment benchmark* showed the improvement of MGPTO over prior techniques, namely STOMP and GPMP2, in finding a collision-free trajectory for a 7 DoF manipulator in cluttered environment. The *torque optimization benchmark* showcased the ability of MGPTO to optimize the sum of absolute robot motor torques during the trajectory and demonstrated improvement in comparison to STOMP. State-of-the-art gradient based methods, such as GPMP2, are unable to optimize a cost function consisting of multiple cost functionals which may be non-differentiable. The *singularity avoidance benchmark* demonstrated the ability of MGPTO to achieve singularity-free motion during the trajectory and demonstrated improvement in comparison to other singularity avoidance methods by relying on the novel geometry-aware index. The qualitative mobile manipulator experiment and the real-world dual-arm robot experiment showed the ability of MGPTO to successfully find solution trajectories for high-DoF robots in cluttered environments.

One may argue that MGPTO should be significantly slower than STOMP due to the fact that MGPTO conducts M trajectory updates at each iteration. However, the computational burden stemming from sampling K trajectories at each iteration is much higher than the gradient estimation procedure in (5.20) and the mean update rule in (5.18). Thus, if we sample the same number of trajectories K with MGPTO and STOMP, a single MGPTO

iteration will be just slightly slower than a single STOMP iteration. On the other hand, some of our mixture component initializations cover parts of trajectory space with smaller cost already in the beginning and MGPTO will therefore often converge in much less iterations, being more sample efficient and faster in practical applications, as shown in the experiments.

When compared to GPMP2, MGPTO showed better exploration of the environment, although being slower. MGPTO successfully found solutions to motion planning problems that GPMP2 was not able to solve. GPMP2 is more appropriate for environments with low complexity, such as the box obstacle scenario, due to its computational efficiency. However, MGPTO outperforms it in complex motion planning problems due to its high capacity for exploration, as well as the ability to explicitly handle joint limits and non-differentiable costs pertaining to arbitrary qualitative metrics or constraints.

Compared to the method [112] presented in Chapter 5, MGPTO achieved similar exploration of the environment while being less computationally demanding. The method presented in Chapter 5 focuses solely on collision avoidance, while MGPTO possesses unique features such as explicit handling of joint position and velocity limits while retaining smoothness, ability to find multiple solutions and the ability to optimize secondary criteria which makes it more suitable for general motion planning problems.

5.6 SUMMARY

In this chapter, we proposed a novel motion planning algorithm dubbed MGPTO that represents trajectories as samples from a mixture of continuous-time GPs and employs stochastic optimization in order to update the MGP parameters in a cost-minimizing manner. The introduced MGP trajectory representation imposes several advantages such as generating smooth trajectories that cover large parts of the trajectory space and enabling the ability to obtain multiple trajectory solutions. The proposed stochastic gradient estimation procedure gives way to enabling optimization of non-differentiable costs and adhering to joint and velocity limits. In conjunction, the MGP representation and the proposed stochastic optimization lead to MGPTO's ability to produce collision-free trajectories while retaining the aforementioned advantages and producing solutions in a computationally efficient manner. We also coupled MGPTO with the novel geometry-aware singularity index to achieve singularity-free motion throughout the obtained trajectories.

MGPTO was evaluated in multiple simulation benchmarks featuring 7 DoF robot arms and a 10 DoF mobile manipulator. Besides collision avoidance, we evaluated its performance in torque minimization and singularity avoidance. We also conducted a real-world experiment with a 14 DoF dual arm robot Rethink Baxter in a laboratory setting. The experimental results demonstrated beyond state-of-the-art performance in collision avoidance, outperforming both GPMP2 [59] and STOMP [71], while keeping a competitive computational efficiency. The results also attest to MGPTO's ability to include different optimization criteria, outperforming several methods in singularity avoidance and successfully optimizing the non-differentiable torque cost.

6

Human-aware motion planning

This chapter deals with the deployment of mobile manipulators in dynamic indoor workspaces, such as a household environment. First, we present an integrated motion planning and perception framework that accounts for the predicted motion of humans in dynamic environments. Then, each component of the framework is dissected with a focus on practical aspects relevant to the real-world application with limited onboard resources. Image segmentation is done with the Mask R-CNN method adapted for detecting humans and estimating their position relative to the robot. For mapping, we rely on the GPU-accelerated composite distance fields which significantly reduce the computation time compared to calculating distance fields from scratch. Furthermore, we propose and implement a novel human trajectory prediction method that combines human intention recognition with trajectory optimization and demonstrates state-of-the-art prediction performance in indoor environments. We also devise a receding-horizon predictive motion planning method based on trajectory optimization that supports real-time whole-body mobile manipulator motion planning. The resultant framework is validated in multiple real-world experiments with a mobile manipulator operating in environments with human presence, using live data from a single robot-mounted RGB-D sensor without any prior knowledge about robot's surroundings. In the end, we propose an alternative trajectory optimization formulation that considers simultaneous robot motion planning and human trajectory prediction.

6.1 Introduction

When robots operate in real-world environments, particularly where humans may co-occupy the workspace, safety is a paramount concern. There is an extensive literature base in the space of autonomous road vehicles for understanding and predicting pedestrian trajectories to assist motion planning and collision avoidance [129, 130, 131, 132]. In contrast, less work has focused on accounting for the predicted trajectories of humans when planning whole-body robot motions in indoor environments, motivating the work presented in this chapter. Compared to the static case, dynamic environments pose many additional challenges that need to be addressed for robots to operate safely and efficiently. To perform tasks in the presence of moving obstacles, motion planning calculations must be performed online and *quickly* for a robot to react to environmental changes that would otherwise result in collisions. For predictive behavior to take place, the robot's perception pipeline must

be continuously updated online so that changes can be perceived sufficiently fast for the motion planning pipeline to react in time.

In this chapter, we propose an integrated framework for predictive whole-body motion planning in dynamic environments. For motion planning, we propose the Receding Horizon And Predictive Gaussian Process Motion Planner 2 (RHAP-GPMP2) – a receding-horizon motion planner that uses composite distance fields to account for the predicted motion of humans in the workspace. We use a state-of-the-art image segmentation method to identify humans and remove dynamic objects from the maintained voxelmap of the static scene. We investigate the task of human motion prediction and propose a planning-based human trajectory prediction method that combines human intention recognition with trajectory optimization. To explore the problem and aid our analysis of the methods, we further produce and release a dataset for human trajectory prediction that includes robot-perspective RGB-D sensor data. We validate our complete framework in hardware experiments and demonstrate effective collision avoidance across multiple scenarios using a trajectory optimization-based approach to whole-body motion planning in the presence of moving obstacles; one such scenario is shown in Figure 6.1. The proposed approach is accompanied by our research paper [133]. Moreover, we also explore an alternative approach that couples human trajectory prediction and robot motion planning into a single optimization problem, simultaneously obtaining trajectory predictions and motion plans.

The key contributions presented in this chapter are

(i) A receding-horizon motion planner that uses composite distance fields to perform time-configuration space motion planning – Receding Horizon And Predictive Gaussian Process Motion Planner 2 (RHAP-GPMP2),

(ii) A novel goal-oriented planning-based human trajectory prediction method that combines human intention recognition with trajectory optimization,

(iii) A comparison of the performance boost provided by GPU-calculated *predicted composite distance fields* with a state-of-the-art algorithm (PBA),

(iv) Experimental verification of our integrated framework on an 8 DoF mobile manipulator in 3D dynamic environments using live sensor data,

(v) Release of the Oxford Indoor Human Motion (Oxford-IHM) dataset which comprises human-motion trajectories in an indoor environment, including motion-capture ground truth trajectories, static RGB-D camera images, and RGB-D data captured from the perspective of a moving robot,

6.2 Related work

In this research, we focus on the development of an integrated framework at the intersection of environment mapping, whole-body motion planning in dynamic environments, and human motion prediction. In the following sections, we review the relevant work across these areas.

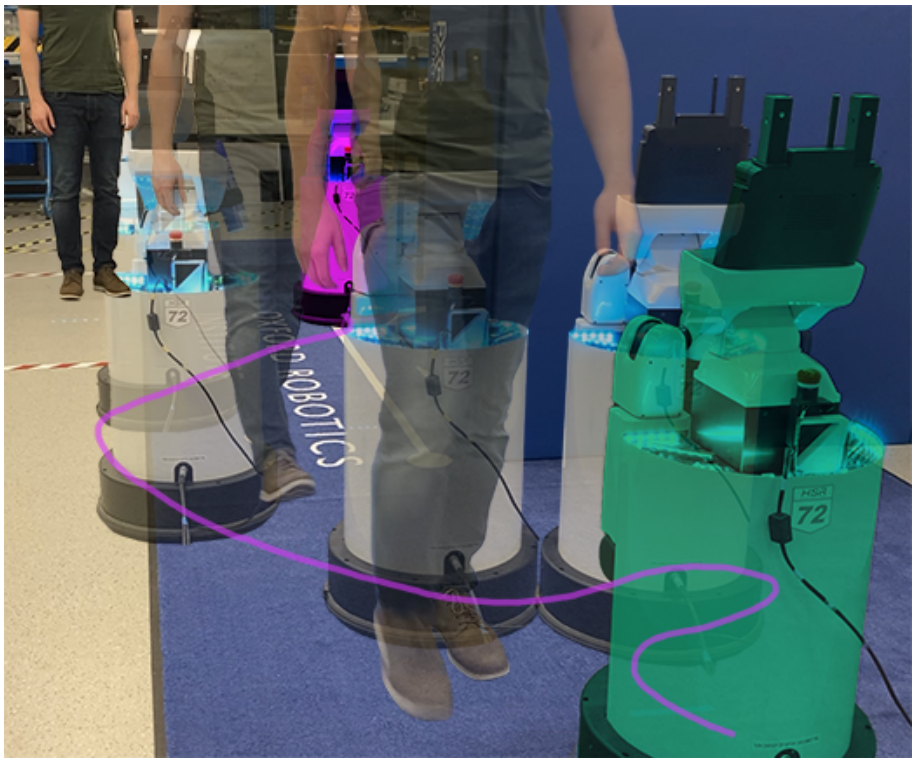


Figure 6.1: A Toyota Human Support Robot (HSR) is given a whole-body goal to place an object on a table at the other end of the room. A wall obstructs the robot's path and, during execution, a person walks towards a goal located behind the robot. Using our proposed approach and accounting for the predicted trajectory of the person, the robot re-plans to pre-emptively move out of the person's path before continuing towards the goal.

6.2.1 *Perception and motion planning in dynamic environments*

Although much of the mapping literature has focused on static environments [134, 135, 136], there have been works that consider dynamic environments. Static-Fusion [137] uses geometric clustering to segment and filter out dynamic obstacles from RGB-D images and fuse observations into a dense static reconstruction of the environment. PoseFusion [138] combines OpenPose [139] with ElasticFusion [136] to reconstruct the static scene while removing humans from the reconstruction. While OpenPose can be used to estimate the positions of body joints of humans within an RGB image, other dynamic scene reconstruction methods consider instance segmentation or optical flow techniques to separate dynamic parts of the scene from the static background. For example, in [140] a feature-based RGB-D Simultaneous Localization and Mapping (SLAM) is performed in dynamic environments using Mask R-CNN [141] image segmentation and optical flow-based motion detection. While their approach demonstrates state-of-the-art localization accuracy, they report an average processing time per frame of 0.42 s and up to 1.10 s when mask inpainting is required. As these numbers show, image segmentation is generally an expensive process and such long computation times may result in behaviors that lack reactivity when deployed on robots in real dynamic environments. Mask R-CNN was similarly used in [142] as a method of detecting potentially dynamic obstacles within a SLAM framework.

To achieve real-time run-rates for SLAM in dynamic environments, MaskFusion [143]

supplements semantic instance segmentation (Mask R-CNN) with geometric segmentation. ReFusion [144] uses a Truncated Signed Distance Field (TSDF) based mapping approach to build static maps of the environment and filter out dynamic objects by using the residuals from the registration and the representation of free space. As with most of the SLAM literature, the aforementioned mapping systems consider the SLAM problem in isolation and do not consider integration with a motion planner. The reverse is also typically true, whereby motion planners in dynamic environments neglect the need for mapping to take place concurrently with live sensing.

ITOMP [51] proposes a parallel optimization approach to motion re-planning in dynamic environments. To perform collision avoidance, the pre-computed Euclidean Distance Transforms (EDTs) are utilized for static obstacle costs, while geometric collision detection is used to assign dynamic obstacle costs. However, the pertaining method was only tested in simulation and neglects consideration of the need to reconstruct the static environment using live sensor data in the presence of dynamic obstacles. In [145], a framework enabling reactive behavior was proposed, using a receding-horizon implementation of GPMP2 [59] in conjunction with the fast GPU-based perception pipeline. While this work enabled reactive whole-body motion planning in response to a dynamic environment, it lacked understanding of dynamic elements in the environment and any reasoning about how they may move in the future. The concept of predicted composite distance fields was introduced in [97] as a fast method of incorporating the predicted motion of moving obstacles directly into the SDF representation of the environment for time-configuration space planning. Using this method, separate environment representations are maintained for each time step in the planned robot trajectory and moving obstacles are propagated using a constant-velocity motion prior. This approach lacks context-aware understanding of human movement which could be used for more realistic trajectory predictions and better predictive motion planning performance.

Voxblox [95] and FIESTA [96] propose incremental mapping frameworks and demonstrate them online. While FIESTA uses a kinodynamic path search method [146], Voxblox is integrated with a trajectory optimization-based motion planner similar to CHOMP [147]. In both cases, only 3D path planning for aerial vehicles is performed rather than whole-body motion planning as we propose herein. GPU-Voxels [94] is a GPU-optimized framework for multiple environment data structures that can be used for collision avoidance. They combine the proposed perception pipeline with a D*-Lite motion planner [148] to demonstrate a mobile robot re-planning in response to newly observed objects. However, they do not demonstrate reactive whole-body behavior in dynamic environments; this is likely due to the curse of dimensionality posed by graph-based motion planners. The method [149] built upon the GPU-Voxels framework to explore fast, exact 3D EDT implementations, such as the Parallel Banding Algorithm (PBA) [150]. They use this work to perform fast motion planning for aerial robots with potential field and wavefront planners, integrated with a GPU-based perception framework that leverages the parallelism in EDT computations.

Of particular interest for our research is the additional aspect of accounting for predicted trajectories. In [151], learned human motions are used to predict the workspace occupancy for usage with STOMP [71] in simulation experiments. The I-Planner algorithm [152] similarly uses offline learning of human actions to generate predicted human motions for

use in motion planning within the workspace of a 7 DoF robot arm. To the best of our knowledge, there does not yet exist a fully integrated perception, motion planning, and prediction pipeline that can enable mobile manipulators to predict the trajectories of moving obstacles and subsequently avoid them in whole-body motion planning tasks. We address this in the work presented in this chapter.

6.2.2 *Human motion prediction*

Motion prediction plays an important role in ensuring the safety of robots and autonomous systems; anticipating how objects will move in a scene enables robots to act in a proactive manner and pre-emptively respond to changes in a dynamic environment to avoid collisions. For inanimate dynamic objects, such as a rolling ball, we can commonly rely on a purely physics-based model, where simple kinematic models (e.g. constant velocity, constant acceleration) often suffice for enabling collision-free robot operation [153]. However, when robots operate in environments alongside humans, safety is of paramount importance and there is a need for more advanced motion prediction to capture the complexity of human behavior. This complexity stems from both internal (goal intent, semantics) and external stimuli (environmental priors, actions of surrounding agents) that influence human motion. The multitude of human motion prediction methods can be categorized by their modelling approach as physics-based, pattern-based and planning-based methods [154].

Physics-based methods predict human motion by propagating the current state through an explicit dynamical model [155, 156, 157, 158, 159, 160]. These methods are typically efficient, interpretable, and work very well for short-term prediction. However, in most cases they do not capture the complexity of the real world, ignoring environmental cues and the possible goals of a person. Notably, a physics-based method that considers a future destination and the surrounding environment was proposed in [157]. However, the therein proposed approach relies on a bird's-eye view and is thus not deployed on a real robot using live sensor data. In [160], human trajectory prediction was integrated with motion planning and perception by using swept-volume based extrapolation on live RGB-D data. However, the obtained motion predictions are used only for stopping a robot's movement when potential collisions are detected, rather than performing motion re-planning and adapting the robot's trajectory.

Pattern-based methods attempt to capture human motion behavior by training function approximators, e.g. neural networks or Gaussian processes, on pre-recorded data [161, 162, 163, 164, 165, 166, 167]. These models have become dominant in recent years due to their performance for long-term prediction in complex, semantically-rich environments. However, they require offline learning with large amounts of training data and offer limited transferability to novel environments due to poor generalization capabilities. A diverse synthetic dataset was generated in [166] to bypass the tedious data collection and a learning framework that exploits scene context to improve generalization was proposed; however, the method is computationally demanding and was not deployed on a real robot. A recurrent neural network is utilized in [167] for encoding short-term dynamics, while environmental constraints are accounted for with trajectory optimization; this method disregards the existence of multiple possible goals and was not demonstrated to work in real environments.

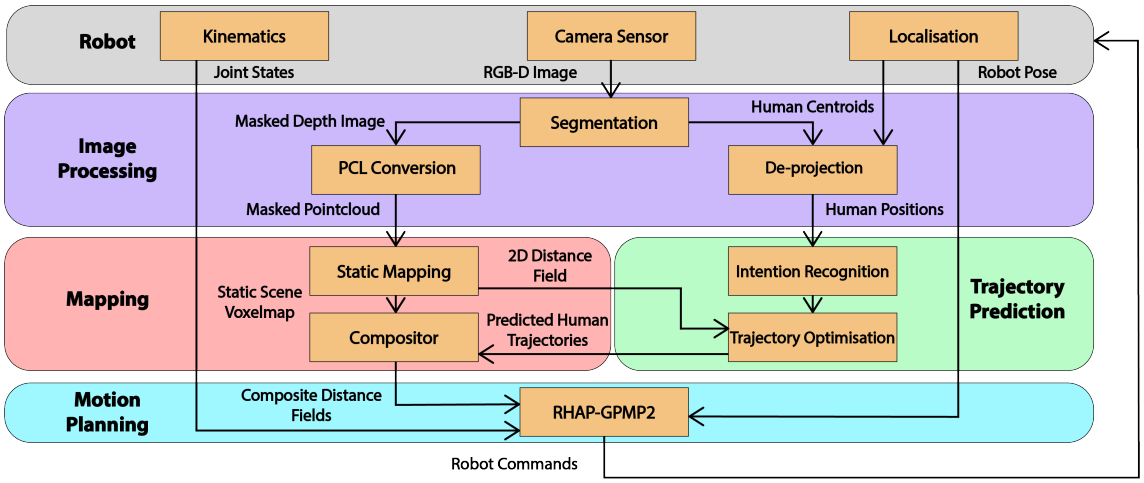


Figure 6.2: Flowchart of our Integrated Framework.

Planning-based approaches assume that a person is moving through an environment towards an existing goal while avoiding obstacles [168, 169, 170, 171, 172, 173, 174]. These approaches offer a good balance between long-term prediction performance and capacity for generalization, but in most cases they require an explicitly defined static map of the environment with the possible goal locations provided, making them difficult to apply on a real robot in an unknown environment. In [172], the authors present a Bayesian framework for intention estimation and use probabilistic roadmaps to obtain trajectory predictions. However, as a consequence of using a sampling-based planning method that does not account for smoothness, the predicted trajectories exhibit rapid changes of direction that are uncharacteristic of humans. Ziebart et al. [168] predict goal-directed behaviors of pedestrians by solving a soft-maximum Markov Decision Process (MDP) with maximum entropy inverse reinforcement learning [175]. They demonstrate real-time robot operation that accounts for human motion prediction but their method has limited generalization capabilities since it relies on learning human reward functions from observed data.

To ensure safe robot operation in indoor environments, we utilize context-specific information and devise a novel trajectory optimization-based method for human motion prediction that respects the underlying dynamical model and environmental cues. Our proposed method, as detailed in Secs. 6.7.1 and 6.7.2, offers a hybrid approach that can be deployed in unknown environments without any prior knowledge but can also incorporate information acquired both offline and online by learning possible human goals from observed data.

6.3 Proposed framework overview

In this section, we outline the modular nature of our proposed framework, as illustrated in Figure 6.2, before examining each module in greater detail. The first part of our proposed framework processes RGB-D images from the camera sensor to extract the information required by other modules. Using a neural network-based instance segmentation method (described in Section 6.5) on the RGB images, we generate masks of objects that are detected in the scene. While our method can be trivially adapted to other identified obstacles, in this

chapter, we focus solely on utilizing the masks of detected humans. If masks are produced above a user-specified threshold score, α_{mask}, we apply the masks to their corresponding depth images; we do this to first extract the estimated position of the humans in the scene, and secondly as a method of removing dynamic obstacles from the scene prior to converting depth images to point clouds. The filtered pointclouds are used to provide live updates to the maintained voxelmap of the static scene, while the extracted positions of people in the scene are passed to the trajectory prediction module.

Using the estimated human positions provided by the image processing module, we perform human trajectory prediction. As described in Section 6.2.2, being able to account for the predicted motion of humans is important for safe robot task execution. While the future trajectory of an inanimate object can often be predicted using constant-velocity or constant-acceleration models, human behavior is more complex and requires a different modelling approach. We propose a hybrid trajectory prediction module (detailed in Section 6.7) that uses a lightweight planning-based approach to perform prediction while retaining the ability to incorporate learned or *prior* information. The proposed trajectory prediction method can rely solely on information obtained from live sensor data, requiring no prior training or initialization. As shown in Figure 6.2, our trajectory prediction module can be divided into *Intention Recognition* (Section 6.7.1) and *Human Trajectory Optimization* (Section 6.7.2).

After predicting trajectories for humans in the scene, we forward these predictions to the mapping module so that the predicted positions of people in the scene can be composited into distance fields that are maintained for each time step in our proposed motion planning algorithm, Receding Horizon And Predictive Gaussian Process Motion Planner 2 (RHAP-GPMP2). With the composite distance fields being continuously updated from the latest observations and trajectory prediction information, our motion planning algorithm is able to re-plan and enable predictive robot behaviors to avoid moving people in the workspace.

6.4 HUMAN MOTION DATASET

For this research, we are concerned with robots that operate in indoor environments co-occupied by humans. In order to evaluate any proposed method for human motion prediction, we require an appropriate dataset that, in line with our ambitions for robot autonomy, contains sensor data from the robot's perspective. While there are a number of publicly available RGB-D datasets, the majority are aimed at applications of SLAM [176, 177], object detection [178, 179], or human activity recognition [180, 181]. Sturm et al. [176] present an RGB-D dataset where sensor data is collected from the robot's perspective for the purpose of evaluating SLAM systems. However, the dataset lacks the presence of moving humans and their ground truth trajectories for which we can try to predict. One of the most relevant dataset for our purposes is the Kinect Tracking Precision (KTP) Dataset [182]. The dataset comprises RGB-D images recorded from a robot's perspective in a scene where humans are moving around while the robot performs locomotion. Totalling only four minutes of video recording, we do not believe that the KTP dataset captures an accurate representation of human motion over a wide enough range of behaviors for prediction purposes. In their recordings, humans move either in a linear or random manner. While this may be how human motion appears sometimes, we believe that humans usually have *intent*, i.e. a desti-



(a) (b) (c)

Figure 6.3: Photos of the marker arrangements used for motion capture tracking of the robot, environment, and humans. Left: A custom 3D-printed frame for mounting markers on the head of an HSR robot. Middle: Multiple objects were tracked and placed in the environment to provide human 'goal' markers. Additional markers were placed at the two entrance/exit locations of the arena. Right: An example setup of the static RGB-D camera within the scene. A custom camera housing was made to attach and calibrate the Vicon markers.

Dataset	Environment	Duration	External Sensors	Robot Perspective	Motion Capture	Goals	Map
KTP [182]	Empty Room	4m 40s	✘	RGB-D	✓	✘	✘
ATC [184]	Shopping Centre	41 days	RGB-D	✘	✘	✓	✘
THÖR [183]	Laboratory	60 mins	3D LiDAR, RGB, Eye Tracking	✘	✓	✓	✓
L-CAS [185]	Office	49 mins	✘	3D LiDAR	✘	✘	✘
MoGaze [186]	Laboratory	180 mins	Eye Tracking	✘	✓	✓	✓
Oxford-IHM	Laboratory/Office	60 mins	RGB-D	RGB-D	✓	✓	✓

Table 6.1: A comparison of publicly available datasets in indoor environments with ground truth human trajectories

nation and task in mind. For example, people will often travel to their desks, a bookcase, or exit through a door in an office environment. Before people enter a room, we could intuitively generate a prior map of where they will go and refine our belief as their trajectory progresses.

The concept of *intent* motivates a dataset in which humans are performing tasks relevant to the environment. The THÖR dataset [183] provides human motion trajectories and broad goal locations within an indoor experiment; however, it uses 3D LiDAR scans from a stationary sensor rather than RGB-D data from a robot perspective. To address the need for a robot perspective RGB-D dataset with task-based human motion trajectories, we propose and release the Oxford Indoor Human Motion (Oxford-IHM) Dataset[1]. We summarize a selection of the most relevant publicly available datasets that include human trajectories, alongside our own, in Table 6.1.

Our dataset was recorded in a large indoor laboratory within which we constructed an arena similar to an office environment. Under a Vicon motion capture setup, we created the arena of interest, measuring 7.1 m×4.2 m, with perimeter walls and two entrance/exit locations. Within the arena, multiple large objects, such as a desk, were arranged in multiple configurations to act both as potential goals and static obstacles. As a tracked human walks between goals in the arena, we recorded RGB-D images from both a static Intel Realsense

[1] Dataset available at https://ori-drs.github.io/oxford-indoor-human-motion-dataset



Figure 6.4: An example map configuration used in the Oxford-IHM dataset. As the Vicon-tracked person walks between goals in the scene, the HSR robot is manually controlled by a human operator to move around the scene and maintain vision of the person.

D435 camera (Figure 6.3c) and a Toyota Human Support Robot's head-mounted ASUS Xtion Pro Live camera. Additionally, we recorded the robot's `tf` data which details the robot's 3D pose and joint transformations over time. While the robot supports an Ethernet connection for data transfer, to avoid trailing cables and maintain recording bandwidth, we opted to control the robot wirelessly and record robot data locally. We used *chrony* time-synchronisation between the robot's onboard clock and an external laptop, which was used to record Vicon marker data and RGB-D image data from the static Realsense camera. During recording, the robot was remotely controlled and navigated around the arena, generally in such a manner that it maintained vision of the tracked person.

The overhead motion capture setup was used to record the ground truth locations of goals, entrance/exit locations, the robot, and the person in the scene. The motion capture arrangement consisted of 18 Vicon Vero 2.2 cameras 2.2 megapixel with 850 nm IR emitters). We calibrated the cameras using a Vicon Active Wand v2 to achieve a sub-millimetre average residual tracking error. For accurate tracking, unique IR marker configurations were affixed to each trackable object. For person tracking, reflective markers were attached to helmets and calibrated to align each helmet's orientation with a person's gaze direction. In the cases of the robot and the external camera, custom mounts were 3D printed (Figure 6.3) and used to calibrate each object's tracked pose with its respective internal camera frame. Our dataset consists of $\approx$ 60 minutes of rosbag data split approximately equally across four different map configurations, each with three runs. For each map configuration, we used the ROS *hector_mapping* package [187] and the robot's onboard Hokuyo UST-20LX laser range sensor to produce a 2D map of the arena. To provide additional variation, our dataset uses two people with the map configurations split equally between them. An example of a

scene from the dataset is shown in Figure 6.4.

6.5 IMAGE PROCESSING

As discussed in Section 6.2.2, numerous approaches have been explored in previous efforts to perform dense environment mapping in dynamic environments [137, 138, 139, 144], with many employing image segmentation techniques [140, 143, 142].

6.5.1 *Image segmentation*

A commonly used state-of-the-art method of image segmentation is Mask R-CNN [141]. Mask R-CNN is an extension of Faster R-CNN [188] that predicts the mask of an object in parallel with bounding box recognition. Image segmentation is typically an expensive operation to perform – in [141] the reported frame rate was 5 Hz on an NVIDIA Tesla M40 GPU. For our purposes of using online environment reconstructions for motion planning in dynamic environments, we require minimal latency between making sensor observations and them being reflected in a motion planner's collision-checking ability. As such, in this work we build on recent advances in image segmentation performance.

While numerous works have sought to improve Mask R-CNN, few works focus on improving the speed of the instance segmentation. CenterMask and CenterMask-Lite [189] are anchor-free one-stage instance segmentation methods that outperform the current state-of-the-art – the authors report that CenterMask-Lite with a VoVNetV2-39 backbone achieves a frame rate of 35 Hz on an NVIDIA Titan Xp GPU. We performed a local benchmarking of the latest release, CenterMask2-Lite, against Mask R-CNN on an RGB-D camera stream and found the lightweight CenterMask2-Lite to run 3.2× faster than Mask R-CNN – 13.4 Hz compared with 4.2 Hz, using the NVIDIA RTX 2060 GPU. Due to the importance of fast perception and motion planning when operating in a dynamic environment, we elected to exploit the enhanced performance provided by CenterMask2-Lite for image segmentation in our perception pipeline.

As described previously, and illustrated in Figure 6.2, we perform image segmentation on a stream of RGB images and use the results for both maintaining the static representation of the environment and for estimating the position of humans in the scene. To perform human position estimation, for each RGB frame that contains masks labelled as a *person* with a score above the specified threshold, α_{mask}, we apply the masks to the corresponding time-synchronised depth image. In this work, we found $\alpha_{mask} = 0.7$ to perform well in consistently masking people even when partially obstructed by obstacles. Using the masked depth image, we extract a depth to associate with the person. In this work, we use the median depth and pixel position of a person's mask to calculate the person's 3D position in the workspace – this position is passed onto the tracking and prediction module.

To carry out object masking and filter out the dynamic element of the scene, we apply valid *person* masks to their corresponding depth images. The filtered depth images are converted to pointclouds for integration into the maintained voxelmap of the static environment. We found that it is beneficial to apply a dilation to the *person* masks before converting to pointclouds. By enlarging the segmentation masks, we reduce leakage from the dynamic

	MOTP (m)		MOTA (%)		FP (%)		FN (%)	
Task	Ours	KTP	Ours	KTP	Ours	KTP	Ours	KTP
Back and Forth	**0.176**	0.196	**91.7**	88.97	5.9	**2.4**	**1.1**	8.5
Random Walk	0.171	0.171	**76.0**	70.93	**2.3**	9.8	**8.6**	18.9
Side-by-Side	0.151	**0.146**	85.9	**87.22**	**0.7**	1.2	**6.7**	11.6
Running	**0.136**	0.143	91.6	**94.57**	**0.0**	1.1	**4.2**	4.4
Group	0.198	**0.181**	**59.4**	47.91	**1.7**	9.1	**15.8**	42.53

Table 6.2: Image Processing and Human Position Estimation Benchmarking. For comparison, the KTP performance metrics are replicated from [182].

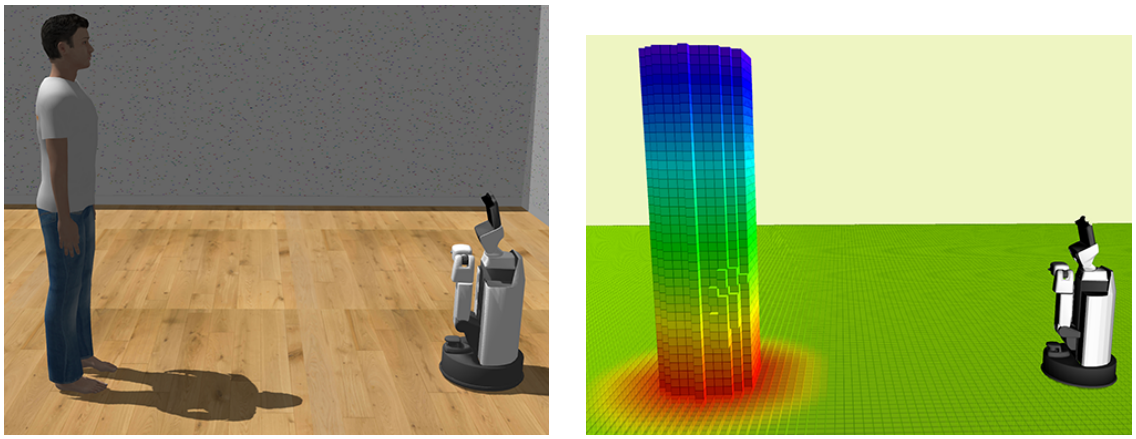
masks into the static voxelmap. We use OpenCV to perform four iterations of dilation with a 5×5 kernel.

To validate our image processing pipeline and demonstrate its effectiveness, we evaluate our method on the KTP Dataset [182], using the same metrics as used in their evaluation [190]. Due to the ground truth positions in this dataset corresponding to a person's tracked head location, in [182] the authors use the centroid of the cluster points belonging to the head of the person and add a 10 cm offset in the viewpoint direction. To provide a similar comparison, rather than using the entire *person* mask for position extraction, we use the top 10% of the mask to correspond with the head. We similarly add an offset in the viewpoint direction and found 12.5 cm to give the best results.

The Multiple Object Tracking Precision (MOTP) metric indicates the ability of a tracker to estimate object positions accurately. The Multiple Object Tracking Accuracy (MOTA) metric indicates the reliability of a tracker to identify objects in an image frame. While we explore instance segmentation in this work, we do not perform tracking between frames – as such, we assume no incorrect object associations in the calculation of the MOTA metric. Additional recorded metrics of interest are the False Positive (FP) and False Negative (FN) rates. Our benchmarking results are presented in Table 6.2. Importantly for this research, we produce similar MOTP values, indicating that our method achieves a similarly competitive accuracy in estimating the position of people in a scene while additionally providing masks for use in the environment reconstruction. In the following sections, we discuss how the extracted positions of people feed into our mapping and trajectory prediction pipelines.

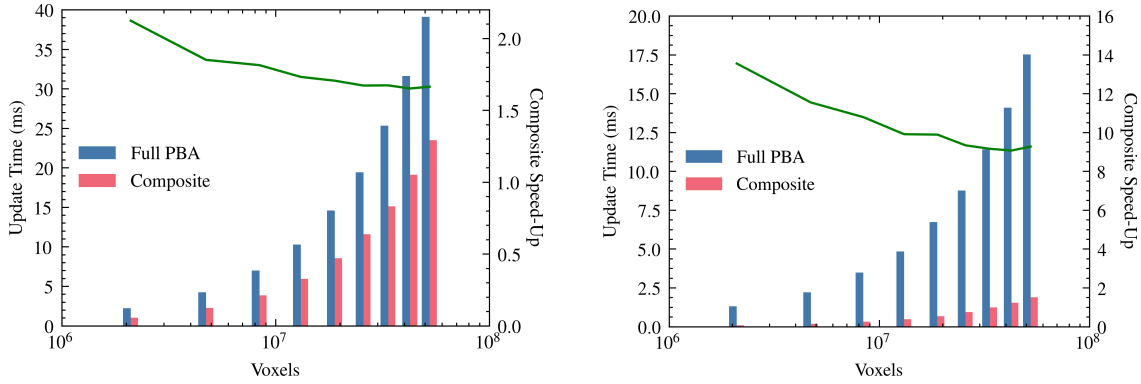
6.6 Mapping - predicted composite distance fields

The early work on *predicted composite distance fields* [97], which carried out the composition on a CPU, suggested that further computational gains could be achieved by performing compositions within a GPU-leveraged framework since the core operation of the method is the *min* operation and thus highly parallelizable. In this section, we explore the gains achievable with such an implementation. Two components are required to generate composite distance fields. Firstly, one needs to maintain a distance field for the environment. Depending on the problem, this may be a static distance field that is computed once at the start of the experiment or continuously updated and maintained as in our framework. Secondly, we require a distance field associated with each (moving) object that is to be *composited* onto the environment distance field. These distance fields can similarly be continuously updated



(a) (b)

Figure 6.5: Top: An example scene in simulation (Gazebo) in which a person is detected by the robot's onboard RGB-D camera. Bottom: The resultant 3D occupancy grid after thresholding the composite distance field for the scene; the distance field of a cylinder is has been composited onto the detected position of the person.



(a) Including Device-Host Transfer Time (b) Excluding Device-Host Transfer Time

Figure 6.6: A comparison of the compute times for composite distance fields and full PBA calculations across a range of voxelmap sizes. The plotted line (green) corresponds to the ratio between the bars, i.e. the resultant speed-up of using composite distance fields. While we see an order of magnitude speed-up in the underlying distance field generation, we find that the device-host transfer time dominates the update time, reducing our overall speed-up from ~ 9.1 – 13.6× to ~ 1.7 – 2.1×.

to represent a live model of the obstacles being tracked. In this work, we are interested in human collision avoidance and so the fine voxelised detail of a human is not necessary; instead, we represent humans with similarly sized cylinder shape primitives. The use of primitive shapes is beneficial since we do not need to be concerned with monitoring the shape of the humans and maintaining a live model; rather, we only need to compute the distance field of the primitive shape once and subsequently track the human positions. However, we note that using shape primitives is a choice in this work rather than a limitation – the distance field could equally be continuously updated to represent an accurate model of a dynamic obstacle as it is observed. There is a large literature base on the dense reconstruction of deformable objects [191, 192].

Given the predicted positions for all dynamic obstacles in the scene for a given time, we can perform a composition of the aforementioned distance fields. This is achieved with a parallel `min` operation between the environment distance field and those of the humans at their predicted positions. An example composition is shown in Figure 6.5. In the case of only considering a single distance field of the environment for each observation update, i.e. no prediction, our method will not be of benefit since only one distance field computation is required. The benefit of our composition approach is apparent when multiple *subsequent distance fields* are required for each environment update loop, such as our motion planning approach as detailed in Section 6.8 which considers multiple predicted distance fields of the environment for each update loop. As such, we perform benchmarking with respect to the calculation time for *subsequent distance fields* against PBA. Benchmarking results are shown in Figure 6.6 where we provide comparisons both including and excluding the time to transfer distance fields from the GPU to the host device. Excluding the transfer times from GPU to host, Figure 6.6b shows our composite method to reduce computation time by 89 – 93 %. Unfortunately, at the short timescales that we achieve, the transfer time becomes a dominant factor, accounting for over 90 % of the overall update time for the composite distance field. However, our composite method still provides a significant performance boost, even after accounting for the transfer time, when compared to a full PBA calculation, cutting the computation time for the resultant distance field by 40 – 53 %.

6.7 HUMAN TRAJECTORY PREDICTION

In indoor environments, a person typically moves towards an intended goal, such as a door to exit through or towards an object to pick up, rather than in a random manner. Therefore, the first component of our trajectory prediction module is *intention recognition* in which we try to determine a person's intended goal.

6.7.1 *Intention recognition*

Suppose there exists a set, $\mathcal{G}$, of K possible goals for a person in the environment, $\boldsymbol{g}_k \in \mathcal{G}$, where a goal is represented by a 2D position vector $\boldsymbol{g}_k = [x_{g_k}, y_{g_k}]^T$. The purpose of the intention recognition module is first to recognise the possible goal locations in the environment that might be of interest to the person, i.e. $\mathcal{G}$, and secondly to identify which of these goals is a person's current intended goal, $\boldsymbol{g}$.

In practice, we believe that $\mathcal{G}$ can be learned over time as objects and areas of interest are observed and identified. Similarly, we believe that we can infer possible goal locations by observing human motion data over time. To explore this idea, we consider a simple *occupancy analysis* method. Given the recorded positions of a person over time, we discretise the positions across a 2D grid of arbitrary size and monitor the number of visitations for each cell, n_i, where the corresponding velocity is less than some threshold, v_{thres}. In the rest of this chapter, we use $v_{\text{thres}} = 0.3\,\text{ms}^{-1}$; this value is much less than the average human walking velocity meaning that grid states which are frequently passed through will not be mislabeled as possible goals. Using a threshold value closer to zero would lead to poor identification of goals where a person is not completely static but has slowed down, e.g.

doors. Using the aforementioned frequency grid, where the most visited cell has $N_{\max}$ visitations, we identify possible goal locations as those cells with $n_i > \frac{N_{\max}}{2}$. If there are multiple adjacent states identified as goals, we take the mean position of those states as a single goal location. By implicitly learning social context cues, our method can also learn additional goals that cannot easily be identified from semantics, e.g. particular gathering points without identifiable objects at those locations. Note that the described goal estimation method can be supplemented with semantic information from the perception pipeline to use identified objects such as desks, sofas, and doors as possible goals, even if they were not visited during the observation time.

We demonstrate our occupancy analysis method on both the Oxford-IHM and THÖR datasets; the results are shown in Figure 6.7 and indicate that the most commonly occupied grid states provide accurate estimates of the ground truth goal locations in each dataset. For both datasets, we analysed segments ~ 5 minutes long. For the THÖR dataset, we tracked all the subjects marked as visitors [183]. On the Oxford-IHM dataset, all ground truth goals were identified, although with an offset due to the ground truth goals being objects that humans maintain a distance from, e.g. a person sits in front of a desk rather than on it. On the THÖR dataset, we identified three out of the five labelled goals; our method did not identify two of the goals for several reasons. Firstly, these goals were in areas with frequent motion capture track drops, a problem that does not occur with live robot sensor data. Secondly, these goals represented exit and entrance locations but without doors; as a result, people passed through without slowing down. One could argue that in this instance, these do not represent accurate goal locations since the people will continue to walk to their true goal locations.

For a given determination of $\mathcal{G}$, we want to determine the probability of each goal, $\boldsymbol{g}_k$, being the human's intended goal, $\boldsymbol{g}$, given that we have observed a history of the person's trajectory, $\boldsymbol{X}_h(t)$, where t is the current time. We use $\boldsymbol{X}_h(t)$ to denote the matrix formed by the past N vector measurements of the human's position, $\boldsymbol{x}_h(t_i) = [x_h(t_i), y_h(t_i)]$ for times $t_i < t$. By framing the human intention recognition problem in this probabilistic manner, we can use Bayes' rule to derive a posterior distribution for goal locations as

$$p\big(\boldsymbol{G}|\boldsymbol{X}_h\big) \propto p(\boldsymbol{G})p\big(\boldsymbol{X}_h|\boldsymbol{G}\big), \tag{6.1}$$

where $p(\boldsymbol{G})$ is a distribution that encodes prior knowledge of goal probabilities and $p\big(\boldsymbol{X}_h|\boldsymbol{G}\big)$ is a conditional distribution that represents the *likelihood* of the recorded past human trajectory for a set of a given goal.

If there is no prior knowledge about the probability distribution for goals, i.e. no goal visitation history, the prior $p(\boldsymbol{G})$ is set as the uniform distribution. On the other hand, if we have an observed (or pre-recorded) history of motion data and perform the occupancy grid analysis described previously, $p(\boldsymbol{G})$ can be set as a categorical distribution where the prior probability for each identified goal is proportional to its number of visitations, N_k:

$$p(\boldsymbol{G} = \boldsymbol{g_k}) = \frac{N_k}{\sum_{l=1}^{K} N_l}. \tag{6.2}$$

Note that in practical applications, the prior goal distribution $p(\boldsymbol{G})$ can be initialised as a uniform distribution when a robot first begins to operate in an environment. As information

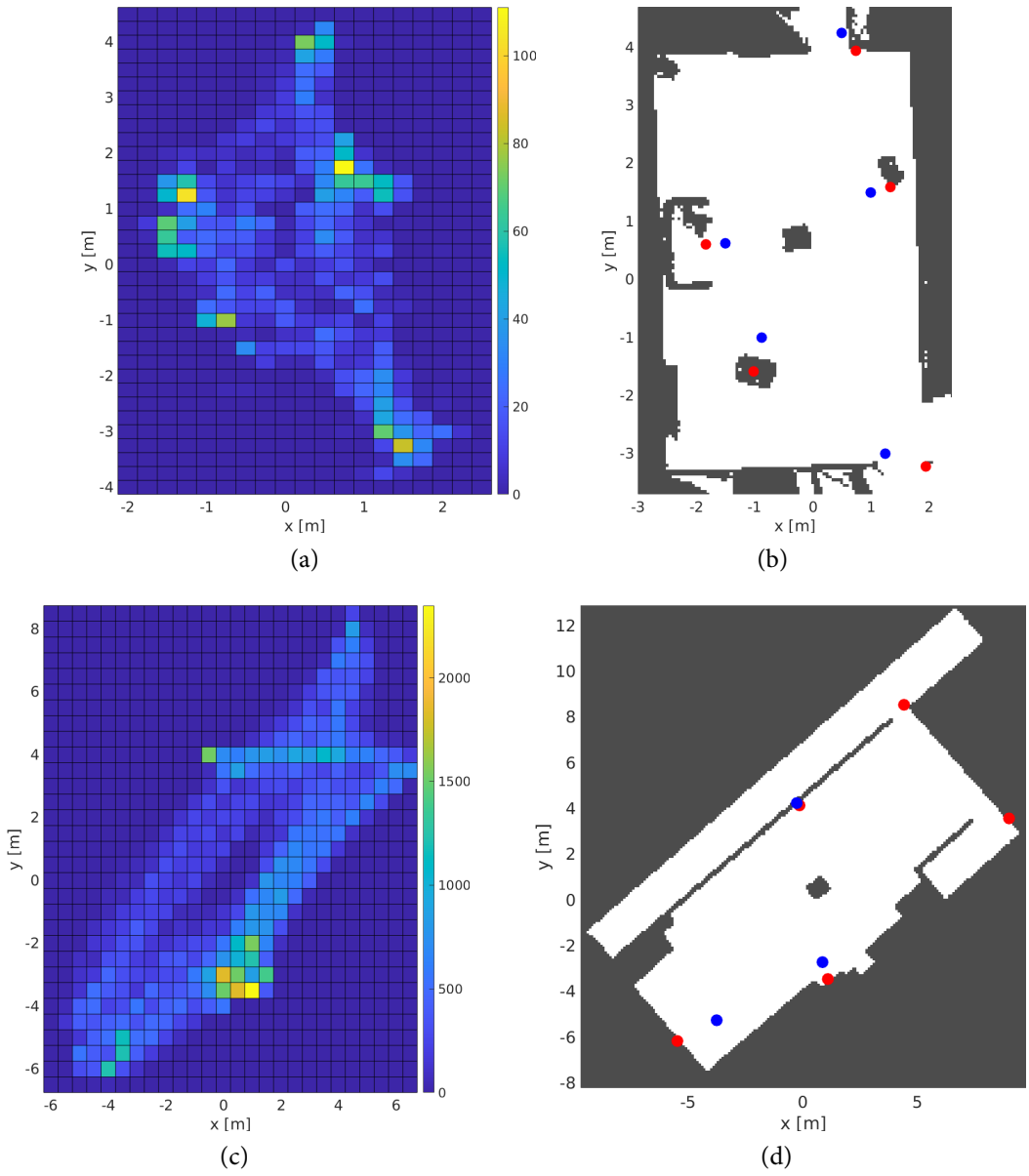


Figure 6.7: By performing our occupancy analysis method on recorded motion data, we can estimate a person's possible goals. We demonstrate this technique using the Oxford-IHM (Figures 6.7a and 6.7b) and THÖR datasets (Figures 6.7c and 6.7d). We monitor the frequency of visitations to each grid state after applying a velocity threshold, resulting in the heatmaps shown. The most occupied states provide a reliable estimate of possible goal locations. Figures 6.7b and 6.7d show maps of example environments from each dataset with the actual (blue) and estimated (red) goal locations.

about the environment and human movement is collected during operation, the prior distribution can be altered online after performing the grid occupancy analysis.

Lastly, we calculate the likelihood, $p\big(\boldsymbol{X}_h(t)|\boldsymbol{G}\big)$, i.e. given the human's recent history, what is the likelihood of each possible goal being the intended one? Intuitively, a person is likely to look at the object they want to reach or move towards in the near future, i.e. the intended goal. As shown by [167, 193], for a set of objects that represents possible goal locations in the environment, a person's gaze is a great predictor of intention. While we

could build on this idea directly and use the difference in angle between a person's gaze and each object to determine the probability of each goal, determining a person's gaze in practice is challenging. Wearable gaze tracking equipment is shown to work very well [167]; however, it is impractical to assume that this is available in everyday applications such as in a household environment. On the other hand, alternative methods that try to estimate the human gaze from images [194, 195] require significant computational resources to work in real-time and have degraded performance when a person is turned away from the robot.

For the aforementioned reasons, we use the human's estimated orientation, obtained from the history of positions, as a predictor of *intent*. While the use of estimated orientation as a motion cue may not be as effective as using a person's gaze, it does not require observability of the human's face and can suffice when other motion cues are unavailable due to hardware constraints. We estimate the human body orientation $\hat{\theta}_h(t_i)$ from the difference between two subsequent positions

$$\hat{\theta}_h(t_i) = \arctan\left(\frac{y_h(t_i) - y_h(t_{i-1})}{x_h(t_i) - x_h(t_{i-1})}\right), \tag{6.3}$$

where we use the 'h' subscript to denote human positions. The relative orientation between a particular goal location and a person is then given by

$$\delta\theta_{g_k}(t_i) = \arctan\left(\frac{y_{g_k} - y_h(t_i)}{x_{g_k} - x_h(t_i)}\right) - \hat{\theta}_h(t_i). \tag{6.4}$$

In practice, there is likely to be noise in individual estimates of the orientation either due to sensor measurements or because a person may briefly look away from their goal without changing their intent. As a result, the relative orientation $\delta\theta_{g_k}(t_i)$ can quickly vary between subsequent timesteps without an actual change in the person's body motion. Therefore we calculate the average over the past N relative orientations

$$\overline{\delta\theta_{g_k}}(t_i) = \frac{1}{N}\sum_{j=0}^{N-1}\delta\theta_{g_k}(t_{i-j}), \tag{6.5}$$

We achieve more stable estimates for the relative orientation by aggregating the recent trajectory history. When the average relative orientation, $\overline{\delta\theta_{g_k}}(t_i)$, is zero, it implies that a person is moving in the direction of the goal $\boldsymbol{g_k}$. Conversely, when $\overline{\delta\theta_{g_k}}(t_i)$ is equal to π, it implies that a person is moving directly away from the goal $\boldsymbol{g_k}$. We thus formulate the likelihood $p\big(\boldsymbol{X}_h|\boldsymbol{G} = \boldsymbol{g_k}\big)$ by calculating the softmax function of the average of past N relative orientations $\overline{\delta\theta_{g_k}}(t_i)$

$$p\big(\boldsymbol{X}_h|\boldsymbol{G} = \boldsymbol{g_k}\big) = \frac{e^{\lambda\overline{\delta\theta_{g_k}}(t_i)}}{\sum_{l=1}^{K} e^{\lambda\overline{\delta\theta_{g_l}}(t_i)}}, \tag{6.6}$$

where λ is a constant that determines the sensitivity of the exponentiated cost. We use $\lambda = 1$ throughout the rest of this work. Following 6.1, the intended goal position is simply extracted by calculating the maximum a posteriori probability (MAP) estimate

$$\hat{\boldsymbol{g}}_{\text{MAP}} = \arg\max_{\boldsymbol{G}\in\mathcal{G}} p(\boldsymbol{G})p\big(\boldsymbol{X}_h|\boldsymbol{G}\big). \tag{6.7}$$

The estimated goal, $\hat{\boldsymbol{g}}_{\text{MAP}}$, has the corresponding probability, $p\big(\boldsymbol{G} = \hat{\boldsymbol{g}}_{\text{MAP}}|\boldsymbol{X}_h\big)$, that represents how sure we are that the estimated goal is the intended one.

6.7.2 *Trajectory optimization*

Once we have determined a person's intended goal, we want to anticipate their motion towards it in order to safely steer the robot away from a person and avoid potential collisions. We make several assumptions about human behaviour in our trajectory prediction method.

Our first assumption is that a person, unobstructed by other factors, will move according to a constant-velocity kinematic model; a person walking directly towards a goal will tend to maintain the same velocity. While we could employ a higher-order kinematic model, such as constant-acceleration, it is unlikely that a person will quickly change their movement speed under normal circumstances, and so a constant-velocity motion model is sufficient for modelling human motion in open spaces [153]. The second assumption that we make is that a person, unlike a moving inanimate object, is generally aware of obstacles in the environment and will try to avoid colliding with them. As such, a robot can use information that it has accumulated about the map of the environment as an environmental prior when predicting a person's trajectory. Our third assumption is that a person is aware of robots operating in the environment and will tend to avoid the space that it occupies.

Using these assumptions, we formulate the human trajectory prediction problem as non-linear trajectory optimization; although primarily used for robot motion planning, in our implementation, we use GPMP2 [59] as a state-of-the-art trajectory optimization method. Since human trajectory prediction and robot motion planning share similarities, we believe that GPMP2, with minor adaptations, is suitable for our prediction problem. In GPMP2, the motion planning problem is framed as a probabilistic inference problem whereby the aim is to formulate the posterior density of a trajectory and solve for the maximum a posteriori (MAP) estimator, just as we did in the previous section. Using Bayes' rule, the posterior distribution of a trajectory, $\boldsymbol{x}$, given the likelihood on a collection of events, $\boldsymbol{e}$, is given by

$$p(\boldsymbol{x}|\boldsymbol{e}) \propto p(\boldsymbol{x})p(\boldsymbol{e}|\boldsymbol{x}), \tag{6.8}$$

where $p(\boldsymbol{x})$ represents the prior that encourages trajectory smoothness, while $p(\boldsymbol{e}|\boldsymbol{x})$ represents the probability of the events $\boldsymbol{e}$ occurring given $\boldsymbol{x}$. In the case of motion planning, $\boldsymbol{e}$ corresponds to binary events that a trajectory $\boldsymbol{x}$ is collision-free at a particular state. Trajectories are represented as samples from a continuous-time GP with a special kernel with the exactly-sparse precision matrix, as described in Section 3.1. Consequently, in [59] the authors show that the probabilistic inference problem in 6.8 can be efficiently solved on a factor graph.

We adopt a similar factor graph formulation and adapt it for human trajectory prediction. Using a structured kernel as in [59], the prior and the likelihood functions can be written as a product of functions

$$\begin{aligned} p(\boldsymbol{x}_h)p(\boldsymbol{e}|\boldsymbol{x}_h) &\propto f^{prior}\left(\boldsymbol{X}_h\right) f^{like}\left(\boldsymbol{X}_h\right) \\ &= \prod_i f_i\left(\boldsymbol{X}_{h,i}\right) \end{aligned} \tag{6.9}$$

where $\boldsymbol{X}_h = \{\boldsymbol{x}_{h,0}, \ldots, \boldsymbol{x}_{h,N}\}$ represents the set of future human positions along the predicted trajectory. The *factors* $\mathcal{F} = \{f_0, \ldots, f_M\}$ are functions that act on variable subsets of the

trajectory. As shown by [66], the posterior distribution can be represented by a bipartite factor graph $G = \{\boldsymbol{X}, \mathcal{F}, \mathcal{E}\}$, where $\mathcal{E}$ is the set of *edges* that connect *variable* and *factor* nodes. To encourage *smoothness* in our predicted human trajectories and account for the tendency to move according to a constant-velocity motion model, we rely on the GP prior in (3.4), given in terms of the mean trajectory $\boldsymbol{\mu}_h$ and covariance $\boldsymbol{\mathcal{K}}_h$. We initialise the mean as a constant-velocity straight line, while the precision matrix is obtained by (3.9). Due to the structured kernel choice, this GP prior has a Markovian structure; as such, it can be written as a product of GP prior factors that depend only on two neighbouring states, $f^{gp}(\boldsymbol{x}_{h,i}, \boldsymbol{x}_{h,i+1})$.

In addition to GP priors that describe how our trajectory behaves, we want to impose knowledge of a person's start and intended goal states. In the context of the human trajectory prediction, the start state is the current position of a person in the environment, while the intended goal state is obtained by our intention recognition method described in Section 6.7.1. We encode start and goal states by using the following factors:

$$f^{start}(\boldsymbol{x}_{h,0}) = \exp\{-\frac{1}{2}\|\boldsymbol{x}_{h,0} - \boldsymbol{x}_{current}\|^2_{\boldsymbol{\Sigma}_{h,0}}\}, \tag{6.10}$$

$$f^{goal}(\boldsymbol{x}_{h,N}) = \exp\{-\frac{1}{2}\|\boldsymbol{x}_{h,N} - \boldsymbol{x}_{goal}\|^2_{\boldsymbol{\Sigma}_{h,N}}\}, \tag{6.11}$$

where N represents the final support state of the trajectory. $\boldsymbol{\Sigma}_{h,0}$ and $\boldsymbol{\Sigma}_{h,N}$ are the isotropic covariance matrices for the start and goal states. Smaller values along the diagonals of these matrices result in higher costs for deviating from the specified start and goal states, encouraging the optimized human trajectory prediction to adhere to the start and goal state constraints.

The motion prior part of the factorization in 6.9 thus becomes a product of factors that takes into account the current position of a person, their intended goal and the constant-velocity movement assumption

$$f^{prior}(\boldsymbol{X}_h) = f^{start}(\boldsymbol{x}_{h,0}) f^{goal}(\boldsymbol{x}_{h,N}) \prod_{i=0}^{N-1} f^{gp}(\boldsymbol{x}_{h,i}, \boldsymbol{x}_{h,i+1}). \tag{6.12}$$

The remaining product of factors represents the likelihood $f^{like}(\boldsymbol{X}_h)$ and encodes all other state-dependent costs and constraints. In the case of human trajectory prediction, we partition it into separate factors that encode collision avoidance with respect to the environment, f^{obs}, and collision avoidance with respect to the moving robot, f^{robot}. The likelihood thus becomes

$$f^{like}(\boldsymbol{X}_h) = \prod_{i=1}^{N-1} f_i^{obs}(\boldsymbol{x}_{h,i}) f_i^{robot}(\boldsymbol{x}_{h,i}). \tag{6.13}$$

For environment collision avoidance factors, we adopt the formulation presented in Section 3.2 which uses a hinge loss function on the SDF of the environment to penalise states that are close to obstacles. As described in Section 6.8, in practice, this SDF is provided by the perception part of our pipeline and updated online; this is in contrast to previous works which pre-compute it [59, 51].

For the robot avoidance factor, f^{robot}, we propose

$$f_i^{robot}(\boldsymbol{x}_{h,i}) = \exp\{-\frac{1}{2}\left\|\boldsymbol{h}(\boldsymbol{x}_{h,i})\right\|^2_{\boldsymbol{\Sigma}_r}\}, \tag{6.14}$$

where $\boldsymbol{h}(\boldsymbol{x}_{h,i})$ is the hinge loss function of the distance between a person and the robot at the current position, $\boldsymbol{x}_r$. The hinge loss function is defined as

$$\boldsymbol{h}(\boldsymbol{x}_{h,i}) = \begin{cases} \varepsilon_r - \left\|\boldsymbol{x}_{h,i} - \boldsymbol{x}_r\right\|_2 & \text{if } \left\|\boldsymbol{x}_{h,i} - \boldsymbol{x}_r\right\|_2 \leq \varepsilon_r \\ 0 & \text{if } \left\|\boldsymbol{x}_{h,i} - \boldsymbol{x}_r\right\|_2 > \varepsilon_r \end{cases}. \tag{6.15}$$

ε_r is a tolerance parameter of our formulation which indicates how close a person is likely to get to a robot before altering their trajectory to avoid collision. If a person is sufficiently far away from the robot, we assume that they will not change their behaviour. However, if the robot comes within the *safety distance*, ε_r, our assumption is that a person will change their behaviour to move away from the robot and avoid collision.

The complete factor graph that we propose for human trajectory prediction can be written as

$$p(\boldsymbol{x_h}|\boldsymbol{e}) \propto f^{start} f^{goal} \prod_{i=0}^{N-1} f_{i,i+1}^{gp} \prod_{i=1}^{N-1} f_i^{obs} f_i^{robot}, \tag{6.16}$$

If we cannot determine a person's intended goal, for instance, if we do not have a set of possible goals or the estimated goal's probability is low, we can omit the goal prior factor. Our proposed prediction method will then work as a constant-velocity model that considers collisions via the robot and obstacle factors. We perform inference on the factor graph using the Levenberg-Marquardt optimization method implemented in GTSAM [121] with an initial damping parameter of 0.01.

6.7.3 *Evaluation*

We evaluate the proposed trajectory prediction method on our human motion dataset described in Section 6.4 and on the THÖR public dataset of human motion trajectories [183]. On both datasets, we predict over four different prediction horizons: 1.6 s, 3.2 s, 4.8 s, and 8.0 s. These prediction horizons cover both short-term and long-term human motion prediction and have previously been used in multiple evaluation pipelines, including the ATLAS benchmark [196]. We thus use them for retaining consistency with the existing body of work in human motion prediction evaluation.

↬ OXFORD-IHM DATASET. Our dataset comprises three different sensor measurements on which we evaluate trajectory prediction performance: the Vicon motion capture data, a static RGB-D camera and an HSR's head-mounted RGB-D camera. The Vicon motion capture data serves as the ground truth against which we compare our predicted trajectories for each type of sensor data. Evaluation of prediction methodologies on the motion capture data gives an indication of their potential performance when given accurate, high-frequency streams of data for the robot's pose, person's pose and goal locations. On the other hand, evaluation of each method on the data obtained using static and robot-mounted RGB-D

cameras indicates their respective prediction performance when operating in real-world environments. Predictions on these data sources account for errors in human position estimation arising from factors such as measurement noise, misdetections, and occlusions.

Further, we compare the performance of our proposed method on the motion capture data with two ablations: (1) without the intention recognition proposed in Section 6.7.1, (2) without the robot avoidance factor proposed in (6.14). These two ablation studies enable us to assess the respective impact of the two features on human motion prediction performance. For each data source, we compare the performance of our proposed method against two baselines: (1) Constant Velocity Model (CVM), (2) Linear Velocity Model (LVM), similar to the ATLAS benchmark [196]. The *CVM* generates predictions by forward propagating the velocity of the person's last observed state, while the *LVM* model generates predictions by forward propagating the observed average velocity of the person. We evaluate trajectory prediction performance using commonly used geometric metrics: Average Displacement Error (ADE) and Final Displacement Error (FDE) [154]. ADE measures the average error across predicted trajectories and the ground truth trajectory, while FDE measures the error of the final predicted point. Since motion capture and RGB -D measurements are inherently asynchronous, for evaluation on RGB-D data sources, we use GP interpolation (described in Section 3.1.2) between states that are temporally closest to the ground truth measurements. For the two baselines methods, we use linear interpolation.

The results of evaluating across all 12 runs in the Oxford-IHM dataset, are shown in Tables 6.3 and 6.4. From the results, our proposed method outperforms the baseline methods for each type of sensor data and every prediction horizon. For the shortest prediction horizon (1.6 s), our proposed method performs similarly to the *CVM* baseline; this is expected since our proposed method has a smoothness factor that is initialised with a constant-velocity motion model. For short prediction horizons, goal and environmental factors have a marginal impact on human motion in absolute terms, meaning that simple kinematic models can often suffice. However, as we predict over longer horizons, our proposed method significantly outperforms the baselines, in line with our expectations, since the *CVM* and *LVM* models do not predict goal-oriented behaviour and disregard environmental cues. Without intent recognition, our method achieves similar performance to the *CVM* and performs significantly worse than our complete proposed method, demonstrating the importance of intention recognition for human motion prediction in indoor environments. Our proposed method performs similarly with and without using the robot avoidance factor, with only minor improvements being achieved with the robot avoidance factor. However, we believe that this factor may become more significant when operating in cluttered environments since it achieves better prediction in specific cases, for example, when the robot blocks a direct path to the intended goal. Figure 6.8 shows an instance of such a situation in the Oxford-IHM dataset.

Using data from the static and robot-mounted RGB -D cameras, we see similar performance trends to those achieved using motion capture measurements, albeit with a greater error. However, the dominant source of this error arises from our position estimation approach applied to the RGB-D images. By comparing the position estimates obtained using our image processing with ground truth measurements, we observe average position estimation errors of 15.9 cm and 30.4 cm respectively for the HSR RGB-D and Static RGB-

		Prediction horizon (s)			
	Method	1.6	3.2	4.8	8.0
Vicon	LVM	0.5 ± 0.02	1.08 ± 0.04	1.7 ± 0.05	2.98 ± 0.07
	CVM	**0.28 ± 0.03**	0.75 ± 0.04	1.35 ± 0.07	2.75 ± 0.11
	Ours	**0.28 ± 0.02**	**0.66 ± 0.06**	**0.99 ± 0.09**	**1.54 ± 0.11**
	w/o Factor	**0.28 ± 0.03**	0.68 ± 0.07	1.04 ± 0.09	1.59 ± 0.12
	w/o Intent	**0.28 ± 0.03**	0.74 ± 0.05	1.25 ± 0.08	2.54 ± 0.12
Static RGB-D	LVM	0.78 ± 0.03	1.38 ± 0.06	2.02 ± 0.09	3.28 ± 0.1
	CVM	**0.59 ± 0.03**	1.03 ± 0.06	1.65 ± 0.1	3.03 ± 0.13
	Ours	**0.59 ± 0.04**	**0.95 ± 0.08**	**1.13 ± 0.1**	**1.86 ± 0.14**
HSR RGB-D	LVM	0.66 ± 0.03	1.26 ± 0.06	1.89 ± 0.09	3.12 ± 0.09
	CVM	0.44 ± 0.03	0.91 ± 0.05	1.54 ± 0.08	2.99 ± 0.10
	Ours	**0.43 ± 0.03**	**0.81 ± 0.07**	**1.13 ± 0.1**	**1.71 ± 0.13**

Table 6.3: Average Displacement Error (ADE) on the Oxford-IHM dataset

		Prediction horizon (s)			
	Method	1.6	3.2	4.8	8.0
Vicon	LVM	1.03 ± 0.03	2.29 ± 0.07	3.6 ± 0.09	6.09 ± 0.18
	CVM	**0.64 ± 0.04**	1.82 ± 0.08	3.26 ± 0.14	6.41 ± 0.2
	Ours	0.65 ± 0.04	**1.36 ± 0.18**	**1.98 ± 0.14**	**2.88 ± 0.15**
	w/o Factor	0.65 ± 0.04	1.38 ± 0.18	2.02 ± 0.14	2.94 ± 0.16
	w/o Intent	0.65 ± 0.04	1.82 ± 0.08	3.2 ± 0.14	6.11 ± 0.21
Static RGB-D	LVM	1.34 ± 0.04	2.6 ± 0.08	3.88 ± 0.11	6.38 ± 0.2
	CVM	**0.92 ± 0.05**	2.1 ± 0.07	3.54 ± 0.13	6.59 ± 0.22
	Ours	**0.92 ± 0.05**	**1.63 ± 0.14**	**2.24 ± 0.16**	**3.13 ± 0.18**
HSR RGB-D	LVM	1.18 ± 0.04	2.45 ± 0.09	3.75 ± 0.09	6.17 ± 0.18
	CVM	**0.80 ± 0.04**	1.98 ± 0.1	3.41 ± 0.16	6.51 ± 0.16
	Ours	0.81 ± 0.04	**1.42 ± 0.13**	**2.12 ± 0.18**	**3.02 ± 0.18**

Table 6.4: Final Displacement Error (FDE) on the Oxford-IHM dataset

D data sources. These results suggest that by further improving the position estimation method, similar trajectory prediction performance can be achieved on RGB-D sources to that achieved using high-frequency motion capture. While the static sensor had fewer measurement drops due to having the whole environment in its field of view, the robot-mounted camera achieved better performance since the robot was usually closer to the person and thus more in accordance with the camera's recommended 'distance of use' for the camera. The results indicate that our proposed human trajectory prediction method works effectively with live sensor data and can be integrated within our proposed perception and motion planning framework as described in previous sections.

THÖR DATASET. We further benchmark our proposed method trajectory prediction on the THÖR dataset [183] in which ten human subjects are tracked in an indoor environment with static obstacles and perform four different social roles that imitate typical activities found in populated spaces (e.g. offices). Enacting these roles results in various motion patterns, and nine out of the ten subjects exhibit goal-oriented behaviour. The

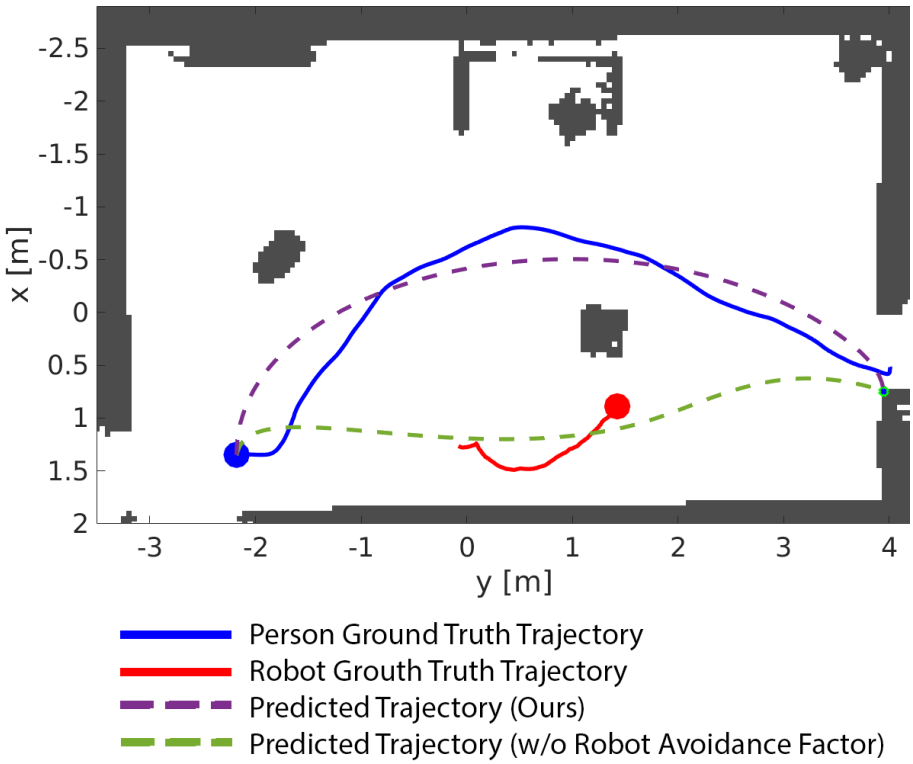


Figure 6.8: An instance in the Oxford-IHM dataset that highlights the impact of using a robot avoidance factor. Without the proposed robot avoidance factor, the predicted human trajectory significantly deviates from the ground truth and collides with the robot.

dataset includes five labelled goals with known ground truth positions. The motion capture data provides ground truth trajectories against which we compare our predictions.

We use the ATLAS benchmark [196] to compare the performance of our proposed method against five different methods, including two baselines (*CVM* and *LVM*). The other three methods are local interaction models, namely the Social force model (*Sof*) [155] and its two predictive extensions *Zan* [158] and *Kara* [159]). These models consider that multiple people are moving in the same environment and will anticipate and evade collisions with each other. As with the Oxford-IHM dataset, we evaluate trajectory prediction performance using the ADE and FDE.

The results of benchmarking for all subjects, across all four runs of the THÖR *One obstacle* experiment, are shown in Table 6.5. In contrast to the Oxford-IHM dataset, the THÖR dataset features fewer obstacles and a larger environment, resulting in more straight-line trajectories with constant velocity. Consequently, we achieve better performance than on the Oxford-IHM dataset. Our method is shown to outperform the baseline methods for all prediction horizons. While the *CVM* baseline achieves a similar level of performance on the shortest prediction horizon, our proposed method significantly outperforms on longer horizons for the reasons explained in Section 6.7.3.1 and in line with our expectations. Our method marginally outperformed the local interaction models (*Sof*, *Zan* and *Kara*) on the ADE metric, but was marginally worse on the FDE metric.

The relatively high standard deviations of the proposed method can be explained by the different social roles assigned to subjects on the THÖR dataset. For the *Lab Worker* and

	Prediction horizon (s)				
	Method	1.6	3.2	4.8	8.0
ADE	CVM	**0.15 ± 0.09**	0.38 ± 0.24	0.71 ± 0.45	1.51 ± 0.91
	LIN	0.29 ± 0.18	0.60 ± 0.38	0.99 ± 0.63	1.84 ± 1.08
	Sof	0.18 ± 0.10	0.36 ± 0.20	0.60 ± 0.35	1.13 ± 0.67
	Zan	**0.15 ± 0.09**	0.34 ± 0.20	0.59 ± 0.36	1.16 ± 0.70
	Kara	0.16 ± 0.08	0.35 ± 0.19	0.60 ± 0.36	1.16 ± 0.69
	Ours	**0.15 ± 0.09**	**0.33 ± 0.26**	**0.57 ± 0.41**	**1.12 ± 0.83**
FDE	CVM	0.28 ± 0.18	0.86 ± 0.54	1.64 ± 1.05	3.54 ± 2.11
	LIN	0.49 ± 0.31	1.20 ± 0.75	2.07 ± 1.30	3.97 ± 2.27
	Sof	0.29 ± 0.16	**0.72 ± 0.42**	**1.27 ± 0.79**	**2.48 ± 1.54**
	Zan	**0.26 ± 0.16**	**0.72 ± 0.43**	1.31 ± 0.82	2.62 ± 1.61
	Kara	0.28 ± 0.15	0.73 ± 0.42	1.31 ± 0.82	2.59 ± 1.59
	Ours	0.28 ± 0.17	0.78 ± 0.64	1.41 ± 0.99	2.98 ± 1.89

Table 6.5: ADE and FDE on the THÖR dataset

Utility Worker roles, the proposed method achieves superior performance in both ADE and FDE because these roles operate in a very goal-oriented manner with no social interactions. In contrast, subjects in the social role, *Visitor*, exhibited behaviours not currently modelled by our method, such as slowing down to interact with other people. For these roles, our proposed method had performed worse than the local interaction models which consider the social component of human behaviour. By not accounting for people slowing down, our proposed method often predicted people travelling further than the ground truth, mainly affecting the FDE performance.

PARAMETERS. Since we use GPMP2 as the backbone for trajectory optimization, our trajectory prediction method depends on a similar set of parameters. Its parameter Q_c specifies the uncertainty in the prior distribution and determines how heavily states are penalised for deviating away from the mean. Σ_{obs} represents the obstacle cost weight with smaller values more strongly penalising collisions with obstacles. Since we introduce the robot avoidance factor in 6.14, we have the additional parameter, Σ_r, that we set to the same value as Σ_{obs} throughout this chapter, equally penalising collisions with the robot and the static environment. The parameter ε represents a *safety distance* from static obstacles. For larger values of ε, optimized trajectories will deviate more from a straight line to maintain a larger distance from obstacles. The proposed robot avoidance factor has a similar parameter, ε_r, indicating a desired *safety distance* from the robot.

For our evaluation, we performed a grid search over parameters Q_c and Σ_{obs} for the Oxford-IHM and THÖR datasets. We found that good trajectory performance was achieved for parameters in the following ranges; $Q_c \in [0.01, 0.5]$ and $\Sigma_{obs} \in [0.02, 0.3]$. We used $Q_c = 0.2$ on Oxford-IHM and $Q_c = 0.05$ on the THÖR dataset. Since ground truth human trajectories on the Oxford-IHM dataset were less smooth than on THÖR due to its environment being smaller and more cluttered, a larger value of Q_c was needed to better account for obstacle avoidance in tight spaces. On both datasets we used $\Sigma_{obs} = 0.1$ and $\varepsilon = 0.4$, resulting in collision-free trajectories for most predictions. Note that ε also considers the radius of a person since the obstacle cost looks at the distance between the *centre* of a person and the

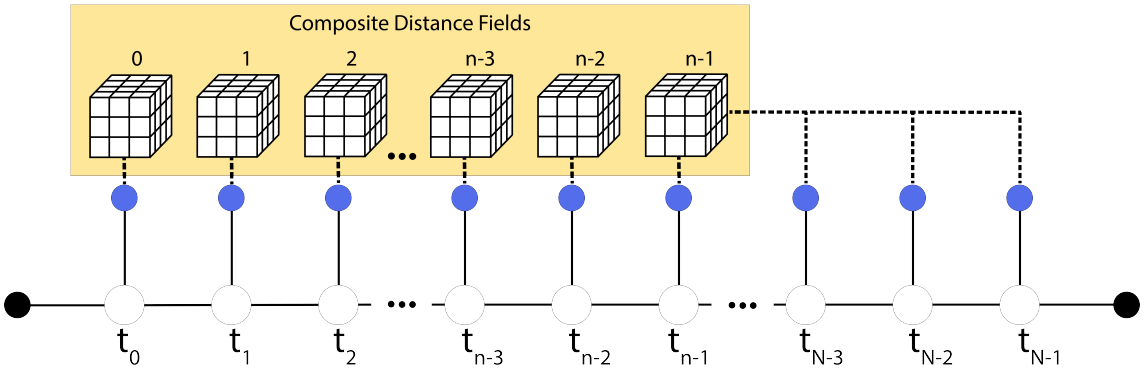


Figure 6.9: The assignment of composite distance fields to the obstacle factors (blue) in RHAP-GPMP2. Given a long time horizon of N timesteps, we assign independent distance fields to the first n timesteps, where n is our dynamic obstacle prediction horizon. For time-indexed obstacle factors greater than n, we assign the n^{th} distance field.

nearest obstacle. The *safety distance* from the robot was set as $\varepsilon_r = 0.8$, double the distance from static obstacles because we account for the robot's radius and also that a person is likely to move farther from a moving robot than a static obstacle. Due to the underlying continuous-time trajectory representation, we must define the total duration of a trajectory, corresponding to an estimation of the time required for a person to reach their intended goal. We estimate this time by calculating the distance between the person's current position and their intended goal and dividing it by their current velocity. For estimated times shorter than the prediction horizon used for evaluation, we use the goal state to predict a person's position for timestamps after the estimated trajectory time.

6.8 Receding horizon and predictive GP motion planner

This section describes how we integrate the methods and concepts discussed in previous sections within a single framework to be deployed on a physical robot. We use the GPU-Voxels framework to maintain a voxelmap of the scene and compute distance fields [94, 149], while motion planning is performed using GPMP2 [59] in a receding-horizon manner, carrying out replanning in a loop. However, we introduce several extensions.

Firstly, as discussed in Section 6.3 and Section 6.5, we introduce image segmentation to remove dynamic obstacles prior to generating pointclouds for integration into the maintained voxelmap of the static scene. At this point, the voxelmap can undergo further filtering if necessary. In the presence of dynamic obstacles, we found it beneficial to filter out voxels that have fewer than five connected voxels; this reduced the instances of spurious voxels being designated as occupied in the voxelmap of the static scene, improving the environment representation accuracy.

Secondly, we propose the Receding Horizon And Predictive Gaussian Process Motion Planner 2 (RHAP-GPMP2). As described in Section 6.7.2, GPMP2 formulates the motion planning problem as probabilistic inference on a factor graph. In RHAP-GPMP2, we continuously monitor the validity of the current trajectory, re-estimate the expected time-to-goal, and re-optimize trajectories to re-evaluate their cost as we observe the environment. If the

current trajectory becomes invalid or a re-optimized trajectory significantly lowers the cost, we generate a new factor graph for trajectory optimization. We use a straight-line trajectory initialisation for the first optimization and in recovery behaviours; otherwise, we re-use and re-optimize the previously planned trajectory to maintain smoothly executed trajectories. In previous work [145], a singular voxelmap was maintained, converted to a distance field, and sent to all obstacle factors within the factor graph used for motion planning. However, RHAP-GPMP2 builds upon the concepts of composite distance fields presented in [97] and extends the therein proposed motion planning method to time-configuration space planning. To achieve this, we maintain:

1. A static voxelmap of the scene
2. A distance field (static or maintained) for each dynamic obstacle in the scene (discussed in Section 6.6)
3. A sequence of n composite distance fields.

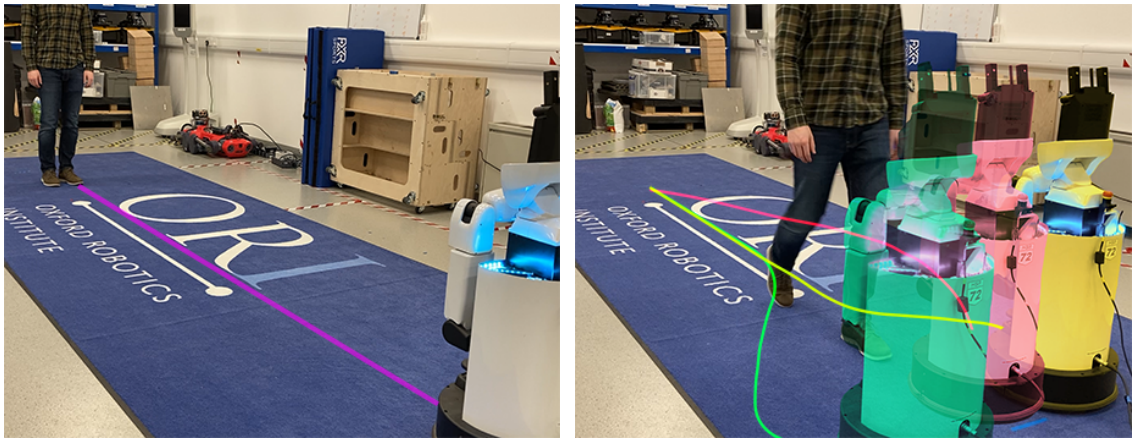
The variable n is determined by how far into the future we wish to incorporate predicted positions for moving objects in the scene. In this work, we use a time-discretisation between factor graph support states of 0.5 s and so choose a value of $n = 20$, corresponding to a time horizon of 10 s. Each time-indexed obstacle factor in the factor graph is associated with a corresponding time-indexed composite distance field. For time-indices greater than n, we assign the composite distance field for time index n. Distance field assignment for RHAP-GPMP2 is illustrated in Figure 6.9.

During each update loop, the static voxelmap is updated using the latest observations of the scene (with the dilated dynamic obstacle masks removed) and composite distance fields are generated using the latest trajectory predictions for dynamic obstacles in the scene, as predicted by our trajectory prediction module described in Section 6.3, Section 6.7.1, and Section 6.7.2. To integrate the human prediction module, we additionally calculate a 2D SDF of the environment by collapsing the maintained 3D voxel grid to 2D and using PBA to calculate the corresponding SDF on the GPU. The SDF is then transferred to the CPU for use in the trajectory prediction module.

6.9 Live hardware experiments

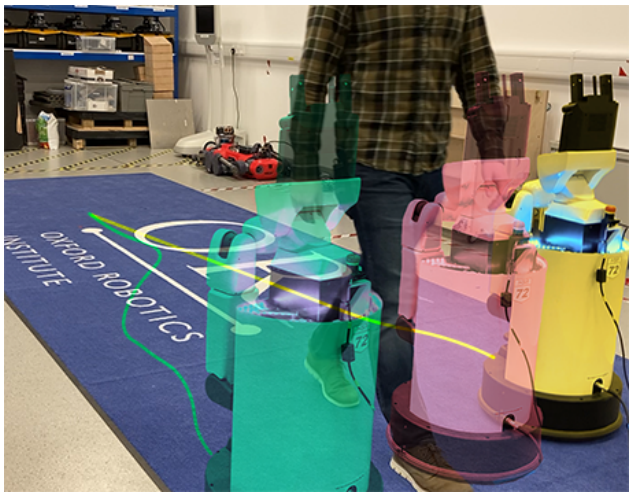
To demonstrate the robustness and capabilities of our whole integrated framework, we deploy our implementation on a physical Toyota Human Support Robot and explore both base-only and whole-body tasks across a range of dynamic scenarios as follows:

1. Change of Places
2. Change of Places with Obstacle
3. Multi-Goal
4. Narrow Passage
5. Change of Places with Half-Wall



(a) Start

(b) *No Prediction* results in collision



(c) Both predictive methods succeed

Figure 6.10: *Change of Places* – the robot is tasked with a base-only goal in front of a person. During execution, the person walks towards a goal behind the robot, requiring the robot to react and move out of the way. All trajectories initially follow a straight line (purple), but as the person approaches, the replanned trajectories for each method diverge as highlighted by the different colours for *No Prediction* (red), *CVM with Prediction* (yellow), *Proposed Method with Prediction* (green). We find that without accounting for the prediction trajectory of the person, the robot collides with the person. In contrast, our proposed framework can avoid collision and complete the task.

Note that in base-only tasks, the motion planner still optimizes in the high-dimensional space of whole-body motions. The baselines of interest for this work are: (1) *No Prediction* [145] and (2) Prediction using *CVM*. In [145], the robot was able to adapt to changes in the environment; however, the human's trajectory was not aimed directly towards the robot in any of the tasks. Hence, the robot was able to perform sufficiently well without prediction. In contrast, for most tasks presented in this work, the robot is required to move out of the way of a person in order to avoid collision. Regarding the baseline implementation, note that in [145] no additional filtering was applied to the maintained voxelmap to remove 'lingering' voxels that a moving obstacle may leave. However, we found that these lingering voxels

provided a substantial disadvantage for motion planning compared to the herein proposed method, which uses dilated segmentation of dynamic obstacles, resulting in a cleaner static voxelmap. As such, to provide an appropriate baseline in our hardware experiments, we introduced the segmentation pipeline to [145] such that the human obstacle is tracked and composited into the singular distance field used for the motion planning.

In our hardware experiments, we perform calculations on an external laptop connected to the HSR via an Ethernet connection to provide sufficient bandwidth for the transfer of images. Hardware specifications for the external laptop are: NVIDIA RTX 2070 Super GPU, 8-core Intel Core i9-10980HK CPU @ 5.30 GHz and 32 GB RAM. Due to the relatively high speed of the human motions in our hardware experiments, we found that the robot always ended up in collision without accounting for the human's predicted motion. We provide supplementary video footage of these experiments[2] and describe the results of each experiment in the following subsections.

6.9.1 *Change of places - prediction is needed*

In our simplest task, *Change of Places*, we do not provide static obstacles, and the robot is tasked with a base-only goal in front of a person. During the task, the person walks towards a goal behind the robot. The resultant task is illustrated in Figure 6.10. We observed that in the *No Prediction* case, without predicting the human's trajectory, re-planned robot trajectories repeatedly become invalid, resulting in collision. Due to the straight-line nature of this task, we found that the *CVM* performed equally as well as our full prediction method. For brevity, in further tasks, we only consider the *CVM* baseline.

6.9.2 *Change of places with obstacle - limitations of the CVM*

With the addition of a central obstacle to the previous task, both the robot and human must take curved geodesics. Results are illustrated in Figure 6.11. This experiment highlighted the limitation of the *CVM* – when a person follows a curved path, the resultant prediction is tangential to the actual path. In this task, this erroneous prediction results in collision. In contrast, our method accurately predicted the person's trajectory, enabling the robot to follow a collision-free trajectory.

6.9.3 *Multi-goal - robust to intention recognition*

While the goal-based aspect of our prediction framework is more extensively evaluated in Section 6.7.3, we provide a hardware experiment in which the person is determined to have two potential goals: one behind the robot's starting position, the other at a hand-wash station across the robot's path. The robot is tasked with a whole-body motion to place a can on the table opposite. Across multiple trials, the robot successfully predicts the person's intended goal and adapts its motions appropriately to execute the task collision-free. Figure 6.12 illustrates these results.

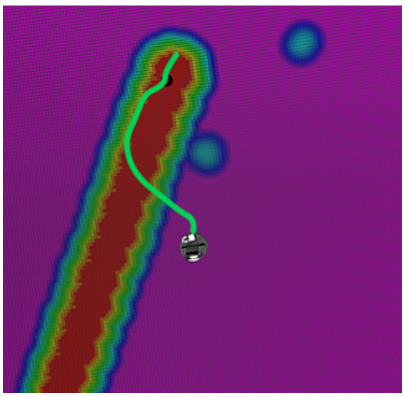
[2] Supplementary video available at `https://youtu.be/gdC3mpZNjG4`

6.9.4 *Additional capabilities*

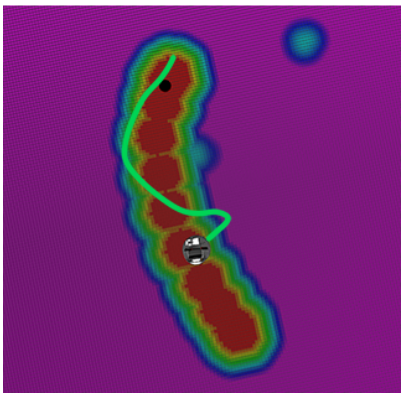
We additionally tested our approach in a *Narrow Passage* task and a variant of the *Change of Places with Obstacle* experiment in which the central obstacle was replaced with a wall across half of the room. In both tasks, the robot successfully adapted to the predicted trajectory of the person and moved to the side of the person's path before continuing towards the goal. Images from the two tasks are shown in Figures 6.1 and 6.13.



(a) *Change of Places with Obstacle*

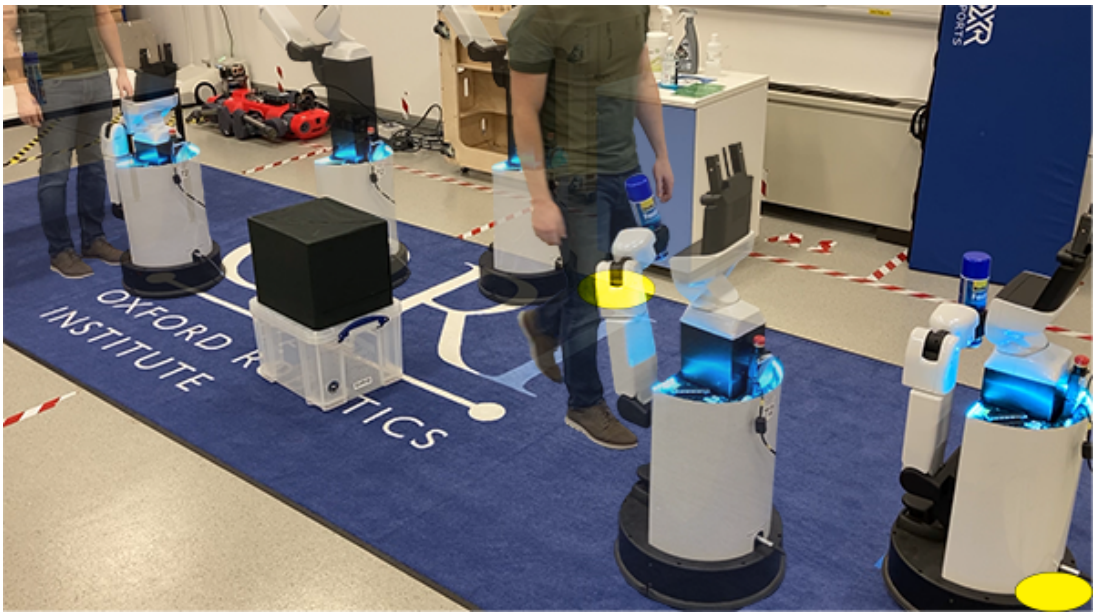


(b) CVM Prediction

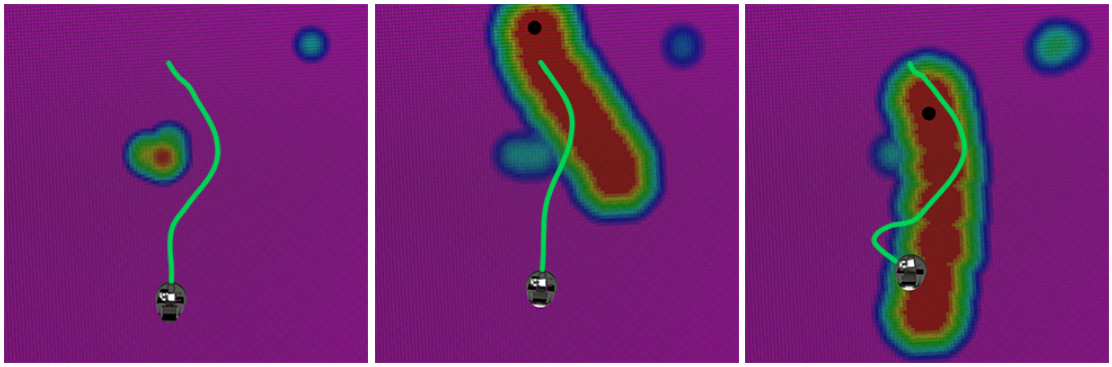


(c) Our Prediction

Figure 6.11: *Change of Places with Obstacle* – a person walks around a static obstacle towards the robot's starting location. Meanwhile, the robot is tasked with a whole-body goal to place a canister on top of a table on the other side of the room. Figure 6.11a shows the trajectories taken by our method (green) and using the *CVM* (yellow). While our methods avoided the person, the *CVM* trajectory did not move out of the way in time, requiring the human to slow down to avoid a collision. The reason for this is explained by Figures 6.11b and 6.11c which show superposed 2D projections of the 3D composite distance fields used for motion planning at two given moments in time. While our prediction method more accurately predicts the human's trajectory, the *CVM* predicts a trajectory tangential to the actual one taken.



(a)



(b) (c) (d)

Figure 6.12: *Multi-Goal* – the robot is tasked with a whole-body goal to place a canister on a table on the other side of a static obstacle. A person in the scene has two possible goal locations - the first is behind the robot's starting location, the second is at a hand-wash station in front of the robot. This experiment demonstrated our framework's ability to adapt and update human trajectory predictions even when the human's intended goal is deemed to have changed. Figure 6.12b shows the initial planned robot trajectory superposed on an aerial view of the ground distance field. Figure 6.12c shows the updated trajectory as the human is deemed to be moving towards the hand-wash station while Figure 6.12c shows the updated trajectory as the prediction module correctly identifies that the person's intended goal is the one behind the robot.

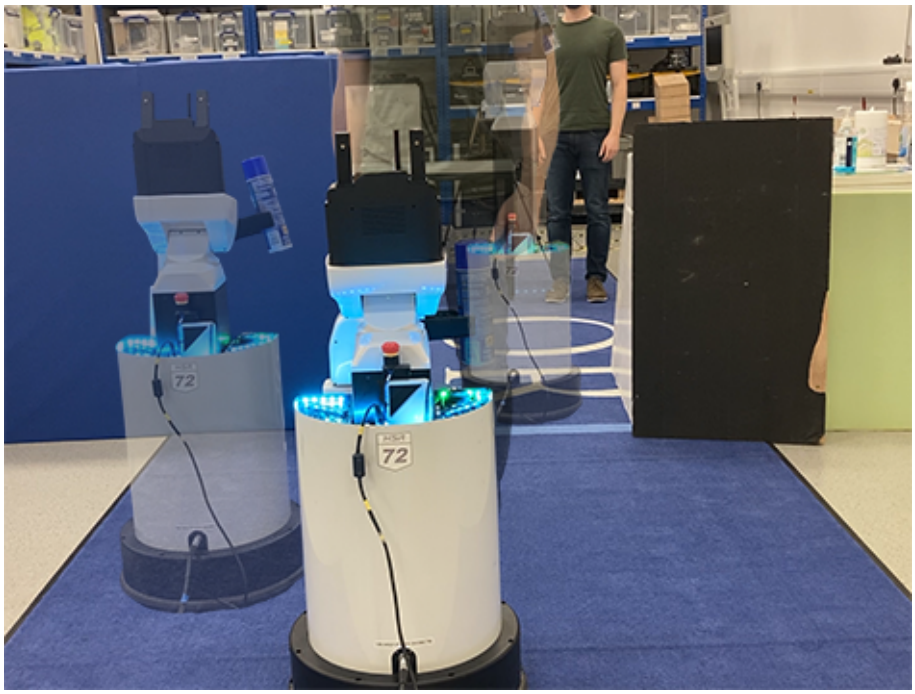


Figure 6.13: *Narrow Passage* – the robot is tasked with a whole-body goal to place a canister on top of a table that is on the other side of a narrow passage. During execution, a person walks through the narrow passage to act as a dynamic obstacle. Our method achieved a successful collision-free trajectory by re-planning to move to the side while the person walked past.

6.10 Simultaneous motion planning and trajectory prediction

The integrated framework presented throughout this chapter has the ability to take into account information about predicted human trajectories within robot motion planning through the use of composite distance fields, as explained in Section 6.8. Conversely, it also has the ability to account for the robot position while predicting the human trajectory through the use of the proposed robot avoidance factor in (6.14). Since both prediction of human trajectories and robot motion planning are carried out as trajectory optimization on a factor graph, naturally, a question arises if the two factor graphs can be coupled to foster the flow of information between them about current and future whereabouts of both the robot and humans in the environment. In this section, we propose a method for simultaneous dynamic obstacle trajectory prediction and robot motion planning as an alternative to the RHAP-GPMP2 method presented in Section 6.8. Since moving obstacles' trajectories are optimized concurrently to motion planning, the proposed approach works in a predictive manner. After computing the initial solution, we use incremental inference for online replanning after an estimate of the moving obstacle position is provided. We also evaluate the proposed method in 2D simulation scenarios and compare it to receding horizon GPMP2. While the results demonstrate that the herein proposed method supports online motion generation in the presence of moving obstacles, generating shorter paths and having faster computation time than receding horizon GPMP2, the integration within the practical framework that supports human-aware whole-body motion planning in real-world

environments remains as future work. The herein proposed method is accompanied by two research papers [197, 198], while the supplementary video of conducted simulations is made available [3].

6.10.1 *Joint factor graph*

The factor graph of a single human trajectory prediction problem was presented in (6.16), and we restate it here with the subscript h denoting factors that are relevant only for the pertaining human trajectory

$$p(\boldsymbol{x_h}|\boldsymbol{e}) \propto f_h^{start} f_h^{goal} \prod_{i=0}^{N-1} f_{h;i,i+1}^{gp} \prod_{i=1}^{N-1} f_{h,i}^{obs} f_i^{robot}. \tag{6.17}$$

The factor graph of a robot motion planning problem solved by GPMP2 [59] is similar; it alo features start and goal states, the robot GP prior and the static obstacle factor. We state the robot motion planning factor graph with the subscript r denoting factors that are relevant only for the robot

$$p(\boldsymbol{x}_r|\boldsymbol{e}) \propto f_r^{start} f_r^{goal} \prod_{i=0}^{N-1} f_{r;i,i+1}^{gp} \prod_{i=1}^{N-1} f_{r,i}^{obs}. \tag{6.18}$$

Inspecting the robot avoidance factor proposed in (6.14), we can see that it relies on positions of both a person and the robot. Clearly, we can leverage this to make a connection between the two graphs. While the factor graph frameworks in (6.17) and (6.18) can be utilized for solving the human trajectory prediction and the robot motion planning independently, we propose combining them into a single larger factor graph, on which the inference can then be performed. We formally define the simultaneous motion planning human trajectory prediction problem as the following factor graph

$$p\left(\boldsymbol{x}_r, \boldsymbol{x}_h^1, \ldots, \boldsymbol{x}_h^M|\boldsymbol{e}\right) = p(\boldsymbol{x}_r|\boldsymbol{e}) \prod_{m=1}^{M} p(\boldsymbol{x}_h^m|\boldsymbol{e}) \tag{6.19}$$

$$\propto f_r^{gp} f_r^{obs} f^{fix} \prod_{m=1}^{M} f_{h,m}^{gp} f_{h,m}^{obs} f_m^{robot}, \tag{6.20}$$

where M is the total number of humans in the environment. For brevity, we denote the GP prior factor of the robot's trajectory as $f_r^{gp} = \prod_{i=0}^{N-1} f_{r;i,i+1}^{gp}$, the GP prior of a person's trajectory as $f_h^{gp} = \prod_{i=0}^{N-1} f_{h;i,i+1}^{gp}$ and the static obstacle factors as $f_r^{obs} = \prod_{i=1}^{N-1} f_{r,i}^{obs}$ and $f_h^{obs} = \prod_{i=1}^{N-1} f_{h,i}^{obs}$. We also use f^{fix} to concisely denote factors for start and goal states of all subjects in the scene. We solve for the full trajectories of the robot and humans simultaneously; the robot's trajectory is planned, while the future human trajectories predicted. A simple illustration of the proposed factor graph is depicted in Fig. 6.14. The major advantage of simultaneous optimization is that it allows direct information flow between the sub-graphs of tracking and planning, which increases performance in both trajectory prediction and planning. It gives our planning algorithm its predictive property.

[3] youtu.be/1l5ErKT1hmc

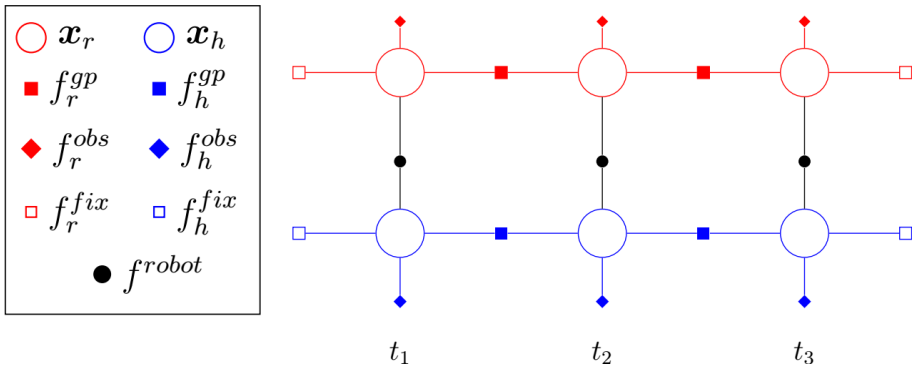


Figure 6.14: A simple illustration of a factor graph for simultaneous prediction of a single human trajectory and mobile robot motion planning with trajectories comprising three states. The red part of this graph corresponds to the GPMP2 graph given in (6.18), while the blue part corresponds to the trajectory prediction graph given in (6.17). These two graphs are connected through factor which allows for predictive motion planning.

Algorithm 3 Simultaneous moving obstacles tracking and motion planning

Initialize trajectories $\boldsymbol{x}_r, \boldsymbol{x}_h^{1\ldots M}$
Create factor graph $p\left(\boldsymbol{x}_r, \boldsymbol{x}_h^1, \ldots, \boldsymbol{x}_h^M | \boldsymbol{e}\right)$
Perform probabilistic inference to obtain MAP trajectories $\boldsymbol{x}_r^*, \boldsymbol{x}_h^{*1\ldots M}$
Convert factor graph to a Bayes tree

for $i = 0 \ldots N-1$ **do**
    if $\boldsymbol{x}_{r,i:i+1}$ is collision-free **then**
        Execute $\boldsymbol{x}_{r,i:i+1}$
        Get moving obstacles' measurements
        Incremental inference update $\boldsymbol{x}_r^*, \boldsymbol{x}_h^{*1\ldots M}$
    else return failure
    end if
end for return success

6.10.2 *Iterative incremental inference*

A major drawback of using nonlinear least squares to solve the factor graph inference is that with every update the problem must be completely resolved which is redundant and inefficient. Even if the factor graph is mostly unaltered, the cost on every factor will be evaluated and every variable updated from scratch. To contend with this problem, [199] proposed efficient updates utilizing the Bayes tree data structure, thus we first convert the factor graph into a Bayes tree [70]. When changes in only a few variables or factors are introduced, only the parts of the Bayes tree related with the changes will be updated, while most of the solution will remain the same. Similarly to [70], we propose incrementally updating the solution of a factor graph in (6.20) which significantly improves the efficiency of inference and achieves real-time performance, as we demonstrate in our experiments. The proposed algorithm is summarized in Algorithm 3, while an example of its execution is presented in Fig. 6.15.

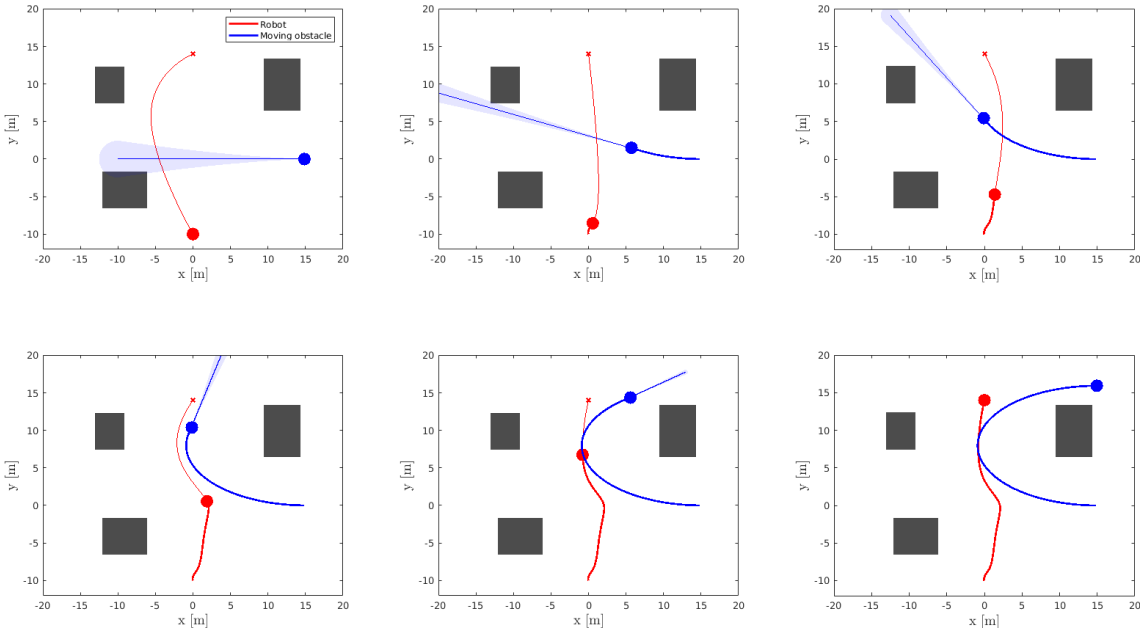


Figure 6.15: An example of simultaneous trajectory prediction and robot motion planning. The red line represents the robot, with the red circle marking its position, the bold line being the executed trajectory and the thin line representing the currently planned future trajectory. The blue line represents a moving obstacle, with other blue elements corresponding to those in red. Blue shading depicts the covariance of estimation and prediction. The obstacle's trajectory is predicted with uncertainty, and replanning is performed at every timestamp.

6.10.3 *Evaluation*

We tested the proposed simultaneous optimization method on two simulation experiments and compared it with a receding-horizon GPMP2. In this evaluation, we consider trajectory prediction of general moving obstacles and not necessarily humans. For this reason, we do not use intention recognition framework presented in Section 6.7.1 and thus do not set goal factors for obstacles, resulting in constant-velocity prediction of their movement. The aim of these simulation benchmarks was to show benefits of the proposed simultaneous optimization for better anticipation of dynamic obstacles and iterative incremental inference for faster replanning, but real-world implementational aspects remain as future work. We first quantitatively demonstrate the improvement of the proposed method over GPMP2 in scenarios where two dynamic obstacles cross the space between the robot and the goal. Then we demonstrate the improvement of the proposed method over GPMP2 in finding a collision-free trajectory in highly cluttered dynamic environment.

Experiments were performed on a system with an Intel Core i7-7700HQ processor and 16 GB of RAM. Both the proposed method and GPMP2 were always initialized with a constant-velocity straight line trajectory. To keep our comparisons fair, in both experiments we used the same parameters for GPMP2 and for our method. We set the total time for every trajectory as t_{total} = 10 s and discretize every trajectory with N = 100 states, meaning that the trajectory is replanned every 100 ms. For GPMP2, we incorporated information about dynamic obstacles inside its signed distance field; future trajectory of obstacles was not

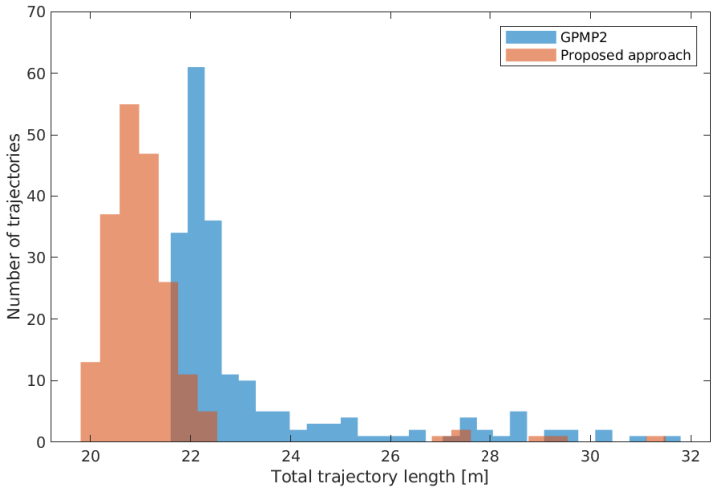


Figure 6.16: Histogram of executed path lengths for the proposed approach and GPMP2.

predicted. While our algorithm employs iterative incremental inference, we rerun GPMP2 from scratch at each time step.

↬ MONTE CARLO SIMULATION OF CROSSING AGENTS. The Monte Carlo simulation of crossing agents consisted of 200 synthetic scenarios with randomized initial conditions of moving obstacles. The robot moves from left to right; it starts at position $\boldsymbol{x}_{start} = [-10, 0]$ m and its goal lies at $\boldsymbol{x}_{goal} = [10, 0]$ m. Two dynamic obstacles cross the space between the robot and the goal, moving with constant velocity, with one obstacle moving from north to south and the other from south to north. The x coordinates of obstacles' start and goal positions were drawn from a uniform distribution on interval $[-10, 10]$ m, while the y coordinates were fixed at 5 m and −5 m.

The proposed method was successful at every scenario and generated robot paths had the average length of 21.71 m, while GPMP2 was successful in 91.5% of scenarios with the average path length 23.32 m. The histograms of the executed path lengths shown in Fig. 6.16 demonstrate that our method more often finds direct paths to the goal than GPMP2. Due to its predictive property, the proposed method is better able to navigate tight spaces and replanned paths do not induce significant direction change as is sometimes the case with GPMP2. Each replanning of the trajectory took 5.1 ms with the proposed method, while it took GPMP2 35.2 ms. An order of magnitude change in computational efficiency is consistent with prior work on incremental inference in a similar setting [200]. The results demonstrate that iterative incremental inference is suitable for real-time application at significantly higher frequencies compared to running probabilistic inference from scratch at each time step.

↬ HIGHLY CLUTTERED ENVIRONMENT. The highly cluttered environment that we designed for qualitative examination of our algorithm consisted of six static and three dynamic obstacles. The robot's position is again initialized as $\boldsymbol{x}_{start} = [-10, 0]$ m with its goal at $\boldsymbol{x}_{goal} = [10, 0]$ m. Every static obstacle is a square with side length $a = 2$ m. The static obstacles were initialized at $[5, -1]$ m, $[0, 8]$ m, $[-7, 4]$ m, $[-4, -2]$ m, $[4, 4]$ m,

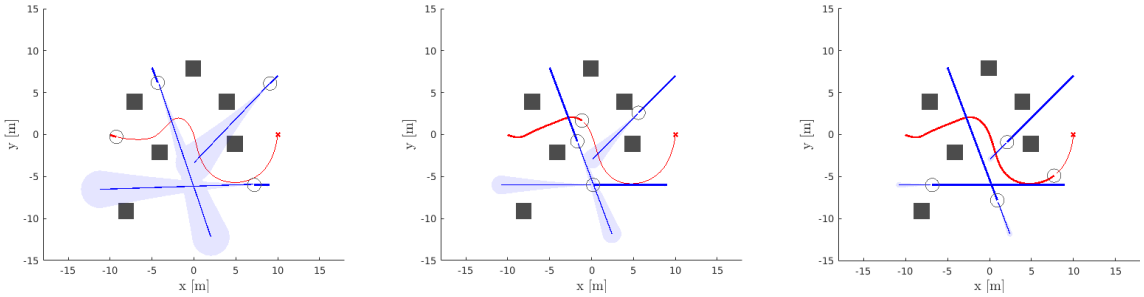


Figure 6.17: The resulting trajectory in the highly cluttered environment experiment. Red line represents the robot, where bold line marks executed trajectory and thin line currently planned future trajectory. Blue lines represents the moving obstacles, bold lines mark estimated trajectories and thin lines predicted future trajectories. Blue shading depicts covariance of estimation and prediction.

$[-8, -9]$ m, respectively. The three dynamic obstacles were initialized at $[10, 7]$ m, $[-5, 8]$ m and $[9, -6]$ m.

The proposed approach successfully navigated the robot through this complex environment, as shown in Fig. 6.17. The executed path length for our approach was 28.48 m, while for GPMP2, which also successfully solved the planning problem, the path length was 34.85 m. Near the first encountered dynamic obstacle, GPMP2 redundantly replanned the trajectory since it had information about robot's path being blocked by an obstacle. This resulted in significantly longer executed path for GPMP2, again demonstrating the benefits of the predictive property of the proposed method. Each replanning of the trajectory with our approach took 4.6 ms on average, while it took GPMP2 46.8 ms, again showing the computational efficiency of iterative incremental inference compared to recomputing probabilistic inference from scratch.

6.11 Additional Remarks

One advantage of the human trajectory prediction approach proposed in this chapter is that it bridges the gap between model-based and learning-based prediction methods. A major limitation of learned models is their ability to transfer to scenarios that differ from those in which it was trained. In contrast, an appealing attribute of our human trajectory prediction method is that it can be readily deployed in any environment by defaulting to a constant-velocity model while retaining the ability to improve over time as a prior distribution is learnt over likely human goal intentions. An interesting direction for further research would be to explore the online-learning of human intentions further and incorporate scene semantics.

In this work, we demonstrated that GPU implementations of predicted composite distance fields can provide a significant performance boost compared to calculating distance fields from scratch. However, as shown in Figure 6.6, the key bottleneck for further composite distance field performance gains is the device-host transfer time. Future work could explore alternative approaches to minimise data transfer between device and host.

While we obtained robust re-planning and collision avoidance behaviours across various tasks, there are several limitations in the presented work that are worth noting. Firstly, we

do not explicitly model uncertainty in our current method of compositing the predicted positions of moving obstacles. Instead, we account for a margin-of-safety via the ϵ parameter within the GPMP2-based obstacle factors – ϵ determines the upper distance used for hinge-loss obstacle costs. While further exploration of this was beyond the scope of the conducted research, we could enlarge the volume of the person/cylinder over the course of the prediction time horizon to appropriately account for a growing uncertainty in future position as time increases.

A natural limitation of our motion planning implementation is that the optimization is prone to get stuck in local minima by only optimizing a single trajectory. While this did not result in collisions in our experiments, the phenomenon is evident in trajectories such as Figures 6.11c and 6.12c – rather than planning to travel on the other side of the static obstacle to the human, the re-planned trajectory avoids the human but stays within the same homotopy class. To address this, one could consider maintaining and optimizing multiple trajectories at a time in different homotopy classes, such as work by [201] and [202], however, this is likely to increase the planning time and limit the robot's ability to react.

It is worth noting that we use two different motion planning approaches in our proposed framework. For robot motion planning, we use predicted versions of the environment for each time step, while for human trajectory prediction, we only use the latest observation of the environment. Our reasoning for this is two-fold; firstly, the walking speed of a human is significantly higher than that of the robot's mobile speed, so a human has less need to account for the predicted trajectory of the robot. Secondly, from our experience, humans will readily travel much closer towards the path of a moving robot, while from the robot perspective, we need to retain a more cautious approach to collision avoidance.

6.12 SUMMARY

In this chapter, we proposed a novel perception, human trajectory prediction and motion planning framework that enables high-dimensional, whole-body motion planning in real-world dynamic environments with human presence. For the perception pipeline, we relied on state-of-the-art image segmentation and 3D mapping methods in order to generate static environment representation and estimate position of humans in the scene in real time. We showcased that a GPU-based implementation of composite distance fields can significantly reduce distance field computation times, underpinning our integrated framework's ability to avoid moving obstacles in real-world environments. We then devised a novel context-aware human trajectory prediction method, specifically aimed at indoor environments, and demonstrated state-of-the-art performance on both a publicly available dataset as well as our own novel goal-oriented dataset, the Oxford Indoor Human Motion (Oxford-IHM) dataset. The motion planning part of our integrated framework employs a receding-horizon motion planner based on continuous-time GPs and trajectory optimization that utilizes predicted composite distance fields to embed the predicted human trajectories in the optimization cost. We verified our proposed framework on a physical mobile manipulator and demonstrated that it is capable of capacitating successful whole-body robot motion planning in a variety of challenging and dynamic real-world environments, while relying on live measurements from a single RGB-D sensor to predict and incorporate the trajectories of humans in a robot's workspace. In the end, we presented an alternative trajectory optimization formulation that simultaneously plans the robot motion and predicts trajectories of moving obstacles, demonstrating potential for future advancement of the proposed integrated framework.

7

Conclusions and outlook

Safely moving through dynamic, unstructured environments and executing tasks that require manipulation of different objects is a daily routine for humans, but still presents a significant challenge for autonomous robots. There exists an apparent gap between robot capabilities required for performing such tasks and robot trajectories that current state-of-the-art motion planning methods can consistently provide in real time. To achieve human-like motion, we require robots with complex kinematic structure that operate in high-dimensional configuration spaces. For the aforementioned reasons, the principal goal of this thesis was to set a foundation that will help in tackling high-dimensional robot motion planning problems in unstructured and dynamic environments, which is a prerequisite to autonomous robots becoming a part of our daily lives.

The thesis started with an overview of existing high-dimensional motion planning methods, providing a structured insight into the three main approaches to tackle the motion planning problem; grid-based, sampling-based and trajectory optimization-based approaches. Relevant methods belonging to each of the three approaches were described in order to further discuss advantages and appropriate use cases for each approach and elucidate reasons behind choosing trajectory optimization approach as a foundation for this thesis' contributions. Trajectory optimization methods start with an initial trajectory that may be infeasible and then optimize it according to a desired state-dependent objective function, endowing robots with ability to generate trajectories that are both feasible and optimal even in high-dimensional configuration spaces. However, they are prone to local minima when the robot structure, environment it operates in or its task-space constraints are particularly complex. Our aim was therefore to devise novel methods that alleviate the local minima problem in practical, real-world applications.

We continued by providing a general background which described mathematical tools and common concepts that formed the basis of proposed motion planning methods based on trajectory optimization. Specifically, we introduced continuous-time GPs as a mathematically well-grounded means to represent continuous trajectories with a small number of discrete states and then use efficient interpolation to query trajectory states at any time of interest, allowing us to circumvent the drawbacks of ubiquitous discrete time representations. We also presented the set-of-spheres as a robot model that is simple to construct and allows for particularly efficient checking of both self-collisions and collisions between the robot's body and the environment it operates in. Moreover, SDFs were introduced as environment representations that support constant-time queries and provide distance information, which

in conjunction with the set-of-spheres model enabled us to devise an obstacle cost function that was used within trajectory optimization for collision avoidance throughout the thesis. We discussed kinematic singularities as a potential hindrance to ensuring robot's agility in case of encountering unforeseeable changes during task execution and presented a novel geometry-aware singularity index that is able to successfully identify singularities even in specific cases in which the commonly-used manipulability and dexterity indices fail.

The first problem that we dealt with considered high-dimensional motion planning in cluttered environments with plethora of local minima. We proposed a stochastic optimization method for trajectory planning, representing robot trajectories as samples from a continuous-time GP. By introducing the heteroscedasticity of the underlying GP, we were able to generate trajectory priors better suited for collision avoidance in motion planning problems, especially in cases when obstacles are position near start or goal robot states. We derived a cross-entropy based derivative-free stochastic optimization method, utilizing importance sampling in order to contend with the local minima problem present in trajectory optimization methods. Due to the sparsity of the employed GP framework, we were able to efficiently estimate the GP covariance at each iteration. We evaluated our method on three simulation scenarios: a maze benchmark, a 7 DoF robot arm planning benchmark and a 10 DoF mobile manipulator trajectory planning example. In a maze benchmark, the proposed method yielded a more thorough exploration of the solution space in complex environments than GPMP2, while having comparable execution time. The robot arm planning benchmark demonstrated the benefits of heteroscedastic GP, while the mobile manipulator planning experiments showed that the proposed method is capable of efficiently tackling high-dimensional trajectory planning problems.

We then focused on multiple-criteria high-dimensional motion planning, with a desire to generate feasible robot trajectories that are also smooth, devoid of singular configurations, and satisfy different arbitrarily chosen objectives. To this end, we proposed a novel motion planning algorithm that represents trajectories as samples from a mixture of continuous-time GPs and employs stochastic optimization in order to update the MGP parameters in a cost-minimizing manner. By introducing the mixture representation and stochastic gradient estimation, the proposed method is able to achieve better exploration of the trajectory space and incorporate non-differentiable optimization criteria. We have also coupled the proposed method with a cost function that penalizes proximity to singular configurations by relying on the novel geometry-aware singularity index based on a Riemannian metric, showcasing reliable singularity avoidance during the whole trajectory. We evaluated our method in multiple simulation scenarios, ranging from a 7 DoF robot arm motion planning, singularity avoidance and torque optimization benchmarks to a 10 DOF mobile manipulator operating in a complex environment, comparing it to state-of-the-art trajectory optimization methods, namely STOMP and GPMP2. The experimental results demonstrated that the proposed method achieves higher success rate and better singularity avoidance than several state-of-the-art methods, while the advantages stemming from MGPs and stochastic optimization, like trajectory smoothness, support of non-differentiable costs, multiple trajectory solutions, and the ability to tackle high-dimensional problems, are inherently kept. Moreover, we conducted a laboratory experiment with a 14 DoF dual arm robot, showcasing the ability of the proposed method to work in a real-world setting.

The third problem we were concerned with pertained to high-dimensional motion planning in real-world dynamic environments with human presence. To enable predictive whole-body motion planning in dynamic environments, we introduced methods for perception, human trajectory prediction and motion planning and integrated them within a novel integrated framework that can account for the predicted trajectories of humans in a scene. We coupled state-of-the-art methods for image segmentation and 3D mapping in order to generate static environment representation and estimate position of humans in the scene in real time. Moreover, we demonstrated the viability and effectiveness of composite distance fields in a GPU-based perception framework and showed that composite distance fields can significantly reduce distance field computation times, underpinning our integrated framework's ability to avoid moving obstacles in real-world environments. We then proposed a novel intention-aware trajectory prediction model for humans in indoor environments and demonstrated state-of-the-art performance on both a publicly available dataset as well as our own novel goal-oriented dataset, the Oxford Indoor Human Motion (Oxford-IHM) dataset, that we make publicly available. For predictive motion planning, we proposed RHAP-GPMP2, a receding-horizon motion planner based on continuous-time GPs and trajectory optimization that utilizes predicted composite distance fields to embed the predicted trajectories of moving obstacles in the objective function. We verified our proposed framework on a physical Toyota Human Support Robot mobile manipulator and demonstrated that our system can use live measurements from a single RGB-D sensor to predict and incorporate the trajectories of humans in a robot's workspace, enabling it to avoid collisions when performing whole-body motion planning across a variety of challenging and dynamic real-world environments. In the end, an alternative trajectory optimization formulation that simultaneously plans the robot motion and predicts trajectories of moving obstacles was presented.

Naturally, the each method proposed in this thesis can still be further improved, extended or applied in different use cases. The strong exploration capacity of the proposed high-dimensional motion planning in cluttered environments could be utilized in future research for finding different homotopy classes in the environment, which could be useful for replanning when a given trajectory becomes infeasible due to changes in the environment. Another possible direction is to further exploit its parallelization capability with a GPU implementation in order to facilitate robot operation in dynamic environments. An interesting avenue for future work would be coupling the proposed method for high-dimensional multi-criteria motion planning with methods for learning from demonstration. The initial parameters of the proposed MGP representation could be learned from human demonstrations on a given task and the proposed stochastic trajectory optimization would enable generalized skill reproduction. Additionally, the robot's mobility gained by staying far from singularities and the inherent ability of the proposed method to find multiple solution trajectories could be exploited in order to adapt and replan robot motion in dynamic environments when a particular trajectory becomes infeasible due to unexpected changes. The proposed method for high-dimensional motion planning in dynamic environments with human presence would benefit from predicting articulated motions of humans, rather than only their 2D position. This could be of avail in human-robot collaboration, enabling more complex robot behavior in shared environments instead of using human

trajectory predictions exclusively for collision avoidance. Furthermore, additional efficiency-improving implementational aspects could be considered, such as relying on the presented simultaneous optimization of predicted robot trajectories and robot motion plans, as well as implementing variable mapping resolution to enable fine-manipulation tasks in more complex environments.

BIBLIOGRAPHY

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

[2] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[3] James Ng and Thomas Bräunl. Performance comparison of bug navigation algorithms. *Journal of Intelligent and Robotic Systems*, 50(1):73–84, 2007.

[4] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[5] Tomás Lozano-Pérez and Michael A Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[6] Colm Ó'Dúnlaing and Chee K Yap. A "retraction" method for planning the motion of a disc. *Journal of Algorithms*, 6(1):104–111, 1985.

[7] Osamu Takahashi and Robert J Schilling. Motion planning in a plane using generalized voronoi diagrams. *IEEE Transactions on Robotics and Automation*, 5(2):143–150, 1989.

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

[9] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3310–3317. IEEE, 1994.

[10] Anthony Stentz et al. The focussed d^* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 95, pages 1652–1659, 1995.

[11] Sven Koenig and Maxim Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005.

[12] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. In *Neural Information Processing Systems (NIPS)*, pages 767–774, 2004.

[13] Benjamin J Cohen, Sachin Chitta, and Maxim Likhachev. Search-based planning for manipulation with motion primitives. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2902–2908. IEEE, 2010.

[14] Mike Phillips, Benjamin J Cohen, Sachin Chitta, and Maxim Likhachev. E-graphs: Bootstrapping planning with experience graphs. In *Robotics: Science and Systems (RSS)*, volume 5, 2012.

[15] Fahad Islam, Venkatraman Narayanan, and Maxim Likhachev. A*-connect: Bounded suboptimal bidirectional heuristic search. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2752–2758. IEEE, 2016.

[16] Wei Du, Fahad Islam, and Maxim Likhachev. Multi-resolution a. In *Annual Symposium on Combinatorial Search*, pages 29–37, 2020.

[17] Shohin Mukherjee, Sandip Aine, and Maxim Likhachev. Mplp: Massively parallelized lazy planning. *IEEE Robotics and Automation Letters*, 2022.

[18] Shohin Mukherjee, Sandip Aine, and Maxim Likhachev. epa* se: Edge-based parallel a* for slow evaluations. *arXiv preprint arXiv:2203.01369*, 2022.

[19] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014.

[20] Steven M LaValle, James J Kuffner, BR Donald, et al. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions*, 5:293–308, 2001.

[21] Jérôme Barraquand and J-C Latombe. A monte-carlo algorithm for path planning with many degrees of freedom. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1712–1717. IEEE, 1990.

[22] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[23] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 995–1001. IEEE, 2000.

[24] James J Kuffner. *Autonomous agents for real-time animation*. PhD thesis, PhD thesis, Stanford University, 1999.

[25] David Hsu. *Randomized single-query motion planning in expansive spaces*. PhD thesis, Stanford University, 2000.

[26] Robert Kindel, David Hsu, J-C Latombe, and Stephen Rock. Kinodynamic motion planning amidst moving obstacles. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 537–543. IEEE, 2000.

[27] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[28] Iram Noreen, Amna Khan, Zulfiqar Habib, et al. Optimal path planning using rrt* based approaches: a survey and future directions. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 7(11):97–107, 2016.

[29] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2997–3004. IEEE, 2014.

[30] Alejandro Perez, Robert Platt, George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2537–2542. IEEE, 2012.

[31] Dustin J Webb and Jur Van Den Berg. Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5054–5061. IEEE, 2013.

[32] David Hsu, J-C Latombe, and Rajeev Motwani. Path planning in expansive configuration spaces. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2719–2726. IEEE, 1997.

[33] Erion Plaku, Kostas E Bekris, Brian Y Chen, Andrew M Ladd, and Lydia E Kavraki. Sampling-based roadmap of trees for parallel motion planning. *IEEE Transactions on Robotics*, 21(4):597–608, 2005.

[34] Pang C Chen and Yong K Hwang. Sandros: a dynamic graph search algorithm for motion planning. *IEEE Transactions on Robotics and Automation*, 14(3):390–403, 1998.

[35] Lydia E Kavraki and Jean-Claude Latombe. Probabilistic roadmaps for robot path planning. *Practical motion planning in robotics: current aproaches and future challenges*, pages 33–53, 1998.

[36] Roland Geraerts and Mark H Overmars. Creating high-quality roadmaps for motion planning in virtual environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4355–4361. IEEE, 2006.

[37] Steven M LaValle, Michael S Branicky, and Stephen R Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8):673–692, 2004.

[38] Michael S Branicky, Steven M LaValle, Kari Olson, and Libo Yang. Quasi-randomized path planning. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1481–1487. IEEE, 2001.

[39] Zaid Tahir, Ahmed H Qureshi, Yasar Ayaz, and Raheel Nawaz. Potentially guided bidirectionalized rrt* for fast optimal path planning in cluttered environments. *Robotics and Autonomous Systems*, 108:13–27, 2018.

[40] Néstor García, Jan Rosell, and Raúl Suárez. Motion planning by demonstration with human-likeness evaluation for dual-arm robots. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11):2298–2307, 2017.

[41] Elon Rimon and Daniel E Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

[42] Charles W Warren. Global path planning using artificial potential fields. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 316–321. IEEE, 1989.

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

[44] Oliver Brock and Oussama Khatib. Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031–1052, 2002.

[45] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: Science and Systems (RSS)*, volume 9, pages 1–10, 2013.

[46] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, 2018.

[47] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 489–494, 2009.

[48] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

[49] Keliang He, Elizabeth Martin, and Matt Zucker. Multigrid chomp with local smoothing. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 315–322. IEEE, 2013.

[50] Arunkumar Byravan, Byron Boots, Siddhartha S Srinivasa, and Dieter Fox. Space-time functional gradient optimization for motion planning. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6499–6506. IEEE, 2014.

[51] Chonhyon Park, Jia Pan, and Dinesh Manocha. Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 207–215, 2012.

[52] Jim Mainprice and Dmitry Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 299–306. IEEE, 2013.

[53] Ixchel G Ramirez-Alpizar, Kensuke Harada, and Eiichi Yoshida. Motion planning for dual-arm assembly of ring-shaped elastic objects. In *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, pages 594–600. IEEE, 2014.

[54] Jiayu Men and Jesús Requena Carrión. A generalization of the chomp algorithm for uav collision-free trajectory generation in unknown dynamic environments. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 96–101. IEEE, 2020.

[55] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[56] Mohamed Elbanhawi, Milan Simic, and Reza Jazar. Randomized bidirectional b-spline parameterization motion planning. *IEEE Transactions on Intelligent Transportation Systems*, 17(2):406–419, 2016.

[57] Zita Marinho, Byron Boots, Anca Dragan, Arunkumar Byravan, Geoffrey J. Gordon, and Siddhartha Srinivasa. Functional gradient motion planning in reproducing kernel hilbert spaces. In *Robotics: Science and Systems (RSS)*, June 2016.

[58] Mustafa Mukadam, Xinyan Yan, and Byron Boots. Gaussian process motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9–15, 2016.

[59] Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion planning as probabilistic inference using gaussian processes and factor graphs. In *Robotics: Science and Systems (RSS)*, volume 12, 2016.

[60] Eric Huang, Mustafa Mukadam, Zhen Liu, and Byron Boots. Motion planning with graph-based trajectories and gaussian process inference. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5591–5598, 2017.

[61] Keshav Kolur, Sahit Chintalapudi, Byron Boots, and Mustafa Mukadam. Online motion planning over multiple homotopy classes with gaussian process inference. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2358–2364. IEEE, 2019.

[62] Mohak Bhardwaj, Byron Boots, and Mustafa Mukadam. Differentiable gaussian process motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 10598–10604. IEEE, 2020.

[63] Tim D Barfoot, Chi Hay Tong, and Simo Särkkä. Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In *Robotics: Science and Systems (RSS)*, volume 10, pages 1–10, 2014.

[64] Marc Toussaint. Robot trajectory optimization using approximate inference. In *International Conference on Machine Learning (ICML)*, pages 1049–1056, 2009.

[65] Marc Toussaint and Christian Goerick. *A Bayesian View on Motor Control and Planning*, pages 227–252. Springer Berlin Heidelberg, 2010.

[66] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[67] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[68] Muhammad Asif Rana, Mustafa Mukadam, S Reza Ahmadzadeh, Sonia Chernova, and Byron Boots. Towards robust skill generalization: Unifying learning from demonstration and motion planning. In *Conference on Robot Learning (CoRL)*, pages 109–118, 2017.

[69] Filip Marić, Oliver Limoyo, Luka Petrović, Trevor Ablett, Ivan Petrović, and Jonathan Kelly. Fast manipulability maximization using continuous-time trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8258–8264. IEEE, 2019.

[70] Mustafa Mukadam, Jing Dong, Frank Dellaert, and Byron Boots. Steap: simultaneous trajectory estimation and planning. *Autonomous Robots*, 43(2):415–434, 2019.

[71] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp:stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574, 2011.

[72] Dmytro Pavlichenko and Sven Behnke. Efficient stochastic multicriteria arm trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4018–4025. IEEE, 2017.

[73] Dylan Jones and Geoffrey A Hollinger. Planning energy-efficient trajectories in strong disturbances. *IEEE Robotics and Automation Letters*, 2(4):2080–2087, 2017.

[74] Michal Dobiš, Martin Dekan, Adam Sojka, Peter Beňo, and František Duchoň. Cartesian constrained stochastic trajectory optimization for motion planning. *Applied Sciences*, 11(24):11712, 2021.

[75] Bahaa Ibraheem Kazem, Ali Ibrahim Mahdi, and Ali Talib Oudah. Motion planning for a robot arm by using genetic algorithm. *Jordan Journal of Mechanical and Industrial Engineering (JJMIE)*, 2(3):131–136, 2008.

[76] Sandi Baressi Šegota, Nikola Anđelić, Ivan Lorencin, Milan Saga, and Zlatan Car. Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms. *International Journal of Advanced Robotic Systems*, 17(2):1729881420908076, 2020.

[77] Jeong-Jung Kim and Ju-Jang Lee. Trajectory optimization with particle swarm optimization for manipulator motion planning. *IEEE Transactions on Industrial Informatics*, 11(3):620–631, 2015.

[78] Takayuki Osa. Multimodal trajectory optimization for motion planning. *The International Journal of Robotics Research*, 39(8):983–1001, 2020.

[79] Marin Kobilarov. Cross-entropy randomized motion planning. In *Robotics: Science and Systems (RSS)*, volume 7, pages 153–160, 2012.

[80] Marin Kobilarov. Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871, 2012.

[81] Timothy D Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.

[82] Peter S Maybeck. *Stochastic models, estimation, and control*. Academic press, 1982.

[83] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[84] Sean Anderson, Timothy D Barfoot, Chi Hay Tong, and Simo Särkkä. Batch nonlinear continuous-time trajectory estimation as exactly sparse gaussian process regression. *Autonomous Robots*, 39(3):221–238, 2015.

[85] Juraj Peršić, Luka Petrović, Ivan Marković, and Ivan Petrović. Spatiotemporal multisensor calibration via gaussian processes moving target tracking. *IEEE Transactions on Robotics*, 37(5):1401–1415, 2021.

[86] Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3859–3866. IEEE, 2012.

[87] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[88] Dieter Fox, Wolfram Burgard, Sebastian Thrun, and Armin B Cremers. A hybrid collision avoidance method for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1238–1243. IEEE, 1998.

[89] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 155–160. IEEE, 2011.

[90] Marija Đakulović, Mijo Čikeš, and Ivan Petrović. Efficient interpolated path planning of mobile robots based on occupancy grid maps. *IFAC Proceedings Volumes*, 45(22):349–354, 2012.

[91] Heinz Breu, Joseph Gil, David Kirkpatrick, and Michael Werman. Linear time euclidean distance transform algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):529–533, 1995.

[92] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136. IEEE, 2011.

[93] René Wagner, Udo Frese, and Berthold Bäuml. 3d modeling, distance and gradient computation for motion planning: A direct gpgpu approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3586–3592. IEEE, 2013.

[94] Andreas Hermann, Florian Drews, Joerg Bauer, Sebastian Klemm, Arne Roennau, and Ruediger Dillmann. Unified gpu voxel collision detection for mobile manipulation planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4154–4160. IEEE, 2014.

[95] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373. IEEE, 2017.

[96] Luxin Han, Fei Gao, Boyu Zhou, and Shaojie Shen. Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4423–4430. IEEE, 2019.

[97] Mark Nicholas Finean, Wolfgang Merkt, and Ioannis Havoutis. Predicted composite signed-distance fields for real-time motion planning in dynamic environments. In *International Conference on Automated Planning and Scheduling (ICAPS)*, volume 31, pages 616–624, 2021.

[98] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, 4(2):3–9, 1985.

[99] Vassilios D Tourassis and Marcelo H Ang Jr. Identification and analysis of robot manipulator singularities. *The International Journal of Robotics Research*, 11(3):248–259, 1992.

[100] Giacomo Marani, Jinhyun Kim, Junku Yuh, and Wan Kyun Chung. A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1973–1978, 2002.

[101] Kévin Dufour and Wael Suleiman. On integrating manipulability index into inverse kinematics solver. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6967–6972, 2017.

[102] Lorenzo Sciavicco and Bruno Siciliano. *Modelling and Control of Robot Manipulators*. Advanced Textbooks in Control and Signal Processing. Springer, 2012.

[103] J Kenneth Salisbury and John J Craig. Articulated hands: Force control and kinematic issues. *The International Journal of Robotics Research*, 1(1):4–17, 1982.

[104] Luis Guilamo, James Kuffner, Koichi Nishiwaki, and Satoshi Kagami. Manipulability optimization for trajectory generation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2017–2022. IEEE, 2006.

[105] Yunong Zhang, Dongsheng Guo, Kene Li, and Jun Li. Manipulability-maximizing self-motion planning and control of redundant manipulators with experimental validation. In *IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1829–1834, 2012.

[106] Jesse Haviland and Peter Corke. A purely-reactive manipulability-maximising motion controller. *arXiv preprint arXiv:2002.11901*, 2020.

[107] Stefano Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, 1997.

[108] Leonel Rozo, Noémie Jaquier, Sylvain Calinon, and Darwin G Caldwell. Learning manipulability ellipsoids for task compatibility in robot manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3183–3189. IEEE, 2017.

[109] Filip Marić, Luka Petrović, Marko Guberina, Jonathan Kelly, and Ivan Petrović. A riemannian metric for geometry-aware singularity avoidance by articulated robots. *Robotics and Autonomous Systems*, 145:103865, 2021.

[110] Noémie Jaquier, Leonel Rozo, Darwin G Caldwell, and Sylvain Calinon. Geometry-aware manipulability learning, tracking, and transfer. *The International Journal of Robotics Research*, 40(2-3):624–650, 2021.

[111] Luka Petrović, Juraj Peršić, Marija Seder, and Ivan Marković. Stochastic optimization for trajectory planning with heteroscedastic gaussian processes. In *European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2019.

[112] Luka Petrović, Juraj Peršić, Marija Seder, and Ivan Marković. Cross-entropy based stochastic optimization of robot trajectories using heteroscedastic continuous-time Gaussian processes. *Robotics and Autonomous Systems*, 133:103618, 2020.

[113] Timothy D Barfoot, James R Forbes, and David J Yoon. Exactly sparse gaussian variational inference with application to derivative-free batch nonlinear state estimation. *The International Journal of Robotics Research*, 39(13):1473–1502, 2020.

[114] F Broussolle. State estimation in power systems: detecting bad data through the sparse inverse matrix method. *IEEE Transactions on Power Apparatus and Systems*, (3):678–682, 1978.

[115] James E Gentle. *Computational statistics*, volume 308. Springer, 2009.

[116] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134:19–67, 2005.

[117] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[118] Brandon Amos and Denis Yarats. The differentiable cross-entropy method. In *International Conference on Machine Learning (ICML)*, pages 291–302. PMLR, 2020.

[119] Chonhyon Park, Jia Pan, and Dinesh Manocha. Real-time optimization-based planning in dynamic environments using gpus. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4090–4097, 2013.

[120] Jing Dong, Mustafa Mukadam, Byron Boots, and Frank Dellaert. Sparse gaussian processes on matrix lie groups: A unified framework for optimizing continuous-time trajectories. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6497–6504. IEEE, 2018.

[121] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical Report GT-RIM-CPR-2012-002, Georgia Institute of Technology, 2012.

[122] David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *STOC*, volume 96, pages 296–303. Citeseer, 1996.

[123] Jonathan D Gammell and Marlin P Strub. Asymptotically optimal sampling-based motion planning methods. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:295–318, 2021.

[124] Luka Petrović, Ivan Marković, and Ivan Petrović. Mixtures of gaussian processes for robot motion planning using stochastic trajectory optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2022.

[125] Luka Petrović, Filip Marić, Ivan Marković, Jonathan Kelly, and Ivan Petrović. Trajectory optimization with geometry-aware singularity avoidance for robot motion planning. In *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pages 1760–1765. IEEE, 2021.

[126] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[127] 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.

[128] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[129] Eike Rehder, Florian Wirth, Martin Lauer, and Christoph Stiller. Pedestrian prediction by planning using deep neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–5. IEEE, 5 2018.

[130] Ivo Batkovic, Mario Zanon, Nils Lubbe, and Paolo Falcone. A computationally efficient model for pedestrian motion prediction. In *European Control Conference (ECC)*, pages 374–379. IEEE, 6 2018.

[131] Carlos Flores, Pierre Merdrignac, Raoul de Charette, Francisco Navas, Vicente Milanes, and Fawzi Nashashibi. A cooperative car-following/emergency braking system with prediction-based pedestrian avoidance capabilities. *IEEE Transactions on Intelligent Transportation Systems*, 20:1837–1846, 5 2019.

[132] Yuanfu Luo, Panpan Cai, Aniket Bera, David Hsu, Wee Sun Lee, and Dinesh Manocha. Porca: Modeling and planning for autonomous driving among many pedestrians. *IEEE Robotics and Automation Letters*, 3:3418–3425, 10 2018.

[133] Mark Nicholas Finean, Luka Petrović, Wolfgang Merkt, Ivan Marković, and Ioannis Havoutis. Motion planning in dynamic environments using context-aware human trajectory prediction. *arXiv preprint arXiv:2201.05058*, 2022.

[134] Richard A Newcombe, Andrew Fitzgibbon, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, and Steve Hodges. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136. IEEE, oct 2011.

[135] Thomas Whelan, Michael Kaess, and Maurice Fallon. Kintinuous: Spatially extended kinectfusion. *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, page 7, 2012.

[136] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. ElasticFusion: Dense SLAM without a pose graph. In *Robotics: Science and Systems (RSS)*, volume 11, 2015.

[137] Raluca Scona, Mariano Jaimez, Yvan R Petillot, Maurice Fallon, and Daniel Cremers. StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3849–3856, 2018.

[138] Tianwei Zhang and Yoshihiko Nakamura. PoseFusion: Dense RGB-D SLAM in Dynamic Human Environments. In *Springer Proceedings in Advanced Robotics*, volume 11, pages 772–780. 2020.

[139] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih En Wei, and Yaser Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021.

[140] Wanfang Xie, Peter Xiaoping Liu, and Minhua Zheng. Moving Object Segmentation and Detection for Robust RGBD-SLAM in Dynamic Environments. *IEEE Transactions on Instrumentation and Measurement*, 70, 2021.

[141] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):386–397, 2020.

[142] Zhongqun Zhang, Jingtao Zhang, and Qirong Tang. Mask R-CNN Based Semantic RGB-D SLAM for Dynamic Scenes. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, volume 2019-July, pages 1151–1156, 2019.

[143] Martin Runz, Maud Buffier, and Lourdes Agapito. MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20, 2019.

[144] Emanuele Palazzolo, Jens Behley, Philipp Lottes, Philippe Giguere, and Cyrill Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7855–7862, 2019.

[145] Mark Nicholas Finean, Wolfgang Merkt, and Ioannis Havoutis. Simultaneous scene reconstruction and whole-body motion planning for safe operation in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3710–3717. IEEE, 2021.

[146] Boyu Zhou, Fei Gao, Luqi Wang, Chuhao Liu, and Shaojie Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4):3529–3536, 2019.

[147] Helen Oleynikova, Michael Burri, Zachary Taylor, Juan Nieto, Roland Siegwart, and Enric Galceran. Continuous-time trajectory optimization for online uav replanning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2016-November, pages 5332–5339. IEEE, 10 2016.

[148] Sven Koenig and Maxim Likhachev. D^* lite. In *AAAI conference on Artificial intelligence*, pages 476–483, 2002.

[149] Christian Juelg, Andreas Hermann, Arne Roennau, and Rüdiger Dillmann. Fast online collision avoidance for mobile service robots through potential fields on 3D environment data processed on GPUs. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2018.

[150] Thanh Tung Cao, Ke Tang, Anis Mohamed, and Tiow Seng Tan. Parallel Banding Algorithm to compute exact distance transform with the GPU. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, pages 83–90, 2010.

[151] Jim Mainprice and Dmitry Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 299–306, 2013.

[152] Jae Sung Park, Chonhyon Park, and Dinesh Manocha. I-Planner: Intention-aware motion planning using learning-based human motion prediction. 38(1):23–39, 2019.

[153] Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll. What the constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters*, 5(2):1696–1703, 2020.

[154] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. 39(8):895–935, 2020.

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

[156] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Transactions on Graphics (TOG)*, 25(3):1160–1168, 2006.

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

[158] Francesco Zanlungo, Tetsushi Ikeda, and Takayuki Kanda. Social force model with explicit collision prediction. *Europhysics Letters (EPL)*, 93(6):68005, 2011.

[159] Ioannis Karamouzas, Peter Heil, Pascal Van Beek, and Mark H Overmars. A predictive collision avoidance model for pedestrian simulation. In *International Workshop on Motion in Games*, pages 41–52. Springer, 2009.

[160] Andreas Hermann, Felix Mauch, Klaus Fischnaller, Sebastian Klemm, Arne Roennau, and Ruediger Dillmann. Anticipate your surroundings: Predictive collision detection between dynamic obstacles and planned robot trajectories on the gpu. In *European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2015.

[161] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[162] Partha Ghosh, Jie Song, Emre Aksan, and Otmar Hilliges. Learning human motion models for long-term predictions. In *International Conference on 3D Vision (3DV)*, pages 458–466. IEEE, 2017.

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

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

[165] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *European Conference on Computer Vision (ECCV)*, pages 507–523. Springer, 2020.

[166] Zhe Cao, Hang Gao, Karttikeya Mangalam, Qi-Zhi Cai, Minh Vo, and Jitendra Malik. Long-term human motion prediction with scene context. In *European Conference on Computer Vision (ECCV)*, pages 387–404. Springer, 2020.

[167] Philipp Kratzer, Marc Toussaint, and Jim Mainprice. Prediction of human full-body movements with motion optimization and recurrent neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1792–1798. IEEE, 2020.

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

[169] Dizan Vasquez. Novel planning-based algorithms for human motion prediction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3317–3322. IEEE, 2016.

[170] Christoph Rösmann, Malte Oeljeklaus, Frank Hoffmann, and Torsten Bertram. Online trajectory prediction and planning for social robot navigation. In *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1255–1260. IEEE, 2017.

[171] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European Conference on Computer Vision (ECCV)*, pages 201–214. Springer, 2012.

[172] Graeme Best and Robert Fitch. Bayesian intention inference for trajectory prediction with an unknown goal destination. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5817–5823. IEEE, 2015.

[173] Andrey Rudenko, Luigi Palmieri, and Kai O Arras. Joint long-term prediction of human motion using a planning-based social force approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4571–4577. IEEE, 2018.

[174] Weiming Zhi, Lionel Ott, and Fabio Ramos. Probabilistic trajectory prediction with structural constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9849–9856. IEEE, 2021.

[175] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[176] Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580, 2012.

[177] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1524–1531. IEEE, 5 2014.

[178] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T. Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3-d object dataset: Putting the kinect to work. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1168–1174. IEEE, 11 2011.

[179] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824. IEEE, 5 2011.

[180] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Unstructured human activity detection from rgbd images. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 842–849. IEEE, 5 2012.

[181] Hao Zhang and L. E. Parker. 4-dimensional local spatio-temporal features for human activity recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2044–2049. IEEE, 9 2011.

[182] Matteo Munaro and Emanuele Menegatti. Fast RGB-D people tracking for service robots. *Autonomous Robots*, 37(3):227–242, 2014.

[183] Andrey Rudenko, Tomasz P Kucner, Chittaranjan S Swaminathan, Ravi T Chadalavada, Kai O Arras, and Achim J Lilienthal. Thör: Human-robot navigation data collection and accurate motion trajectories dataset. *IEEE Robotics and Automation Letters*, 5(2):676–682, 2020.

[184] Drazen Brscic, Takayuki Kanda, Tetsushi Ikeda, and Takahiro Miyashita. Person tracking in large public spaces using 3-d range sensors. *IEEE Transactions on Human-Machine Systems*, 43:522–534, 11 2013.

[185] Zhi Yan, Tom Duckett, and Nicola Bellotto. Online learning for human classification in 3d lidar-based tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 864–871. IEEE, 2017.

[186] Philipp Kratzer, Simon Bihlmaier, Niteesh Balachandra Midlagajni, Rohit Prakash, Marc Toussaint, and Jim Mainprice. Mogaze: A dataset of full-body motions that includes workspace geometry and eye-gaze. *IEEE Robotics and Automation Letters*, 6(2):367–373, 2020.

[187] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 155–160. IEEE, 2011.

[188] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 6 2017.

[189] Youngwan Lee and Jongyoul Park. CenterMask: Real-time anchor-free instance segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13903–13912, 2020.

[190] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[191] Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon, and Shahram Izadi. 3d scanning deformable objects with a single rgbd sensor. volume 07-12-June-2015, pages 493–501. IEEE, 6 2015.

[192] Rui Yu, Chris Russell, Neill D. F. Campbell, and Lourdes Agapito. Direct, dense, and deformable: Template-based non-rigid 3d reconstruction from rgb video. pages 918–926. IEEE, 12 2015.

[193] Tomislav Petković, Luka Petrović, Ivan Marković, and Ivan Petrović. Ensemble of lstms and feature selection for human action prediction. In *Intelligent Autonomous Systems (IAS) 16*, pages 429–441, Cham, 2022. Springer International Publishing.

[194] Petr Kellnhofer, Adria Recasens, Simon Stent, Wojciech Matusik, and Antonio Torralba. Gaze360: Physically unconstrained gaze estimation in the wild. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6912–6921, 2019.

[195] Seonwook Park, Adrian Spurr, and Otmar Hilliges. Deep pictorial gaze estimation. In *European Conference on Computer Vision (ECCV)*, pages 721–738, 2018.

[196] Andrey Rudenko, Wanting Huang, Luigi Palmieri, Kai O Arras, and Achim J Lilienthal. Atlas: a benchmarking tool for human motion prediction algorithms. In *Robotics: Science and Systems (RSS) Workshop on Social Robot Navigation*, 2021.

[197] Luka Petrović, Ivan Marković, and Marija Seder. Multi-agent gaussian process motion planning via probabilistic inference. *IFAC-PapersOnLine*, 51(22):160–165, 2018.

[198] Luka Petrović, Filip Marić, Ivan Marković, and Ivan Petrović. Gaussian processes incremental inference for mobile robots dynamic planning. *IFAC-PapersOnLine*, 53(2):9584–9589, 2020.

[199] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[200] Xinyan Yan, Vadim Indelman, and Byron Boots. Incremental sparse gp regression for continuous-time trajectory estimation and mapping. *Robotics and Autonomous Systems*, 87:120–132, 2017.

[201] Keshav Kolur, Sahit Chintalapudi, Byron Boots, and Mustafa Mukadam. Online motion planning over multiple homotopy classes with gaussian process inference. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2358–2364, 11 2019.

[202] Wolfgang Xaver Merkt, Vladimir Ivan, Traiko Dinev, Ioannis Havoutis, and Sethu Vijayakumar. Memory clustering using persistent homology for multimodality- and discontinuity-sensitive learning of optimal control warm-starts. *IEEE Transactions on Robotics*, 37(5):1649–1660, 2021.

CURRICULUM VITAE

Luka Petrović was born in Zagreb, Croatia in 1995. He received his B.Sc. and M.Sc. degrees in electrical engineering and information technology from the University of Zagreb Faculty of Electrical Engineering and Computing (FER) in 2015 and 2017, respectively. As a recognition for his outstanding academic achievement during graduate studies, he received the Dean's Award "Josip Lončar" in 2016 and the "Bronze plaque Josip Lončar" in 2017, both from FER. He also received the Rector's award for remarkable student research work titled "Decentralized control of the multi-agent robotic system" and a scolarship for excellence from the University of Zagreb.

In 2017, upon finishing his studies, he joined FER and the Department of Control and Computer Engineering as a researcher on the project "DATACROSS – Advanced Methods and Technologies in Data Science and Cooperative Systems", funded by the European Union (EU). He also started his PhD studies at FER in the same year. Furthermore, he actively participated in the EU funded project "L4MS – Logistics for Manufacturing SMEs", where he worked on cyber-physical middleware (2018-2019). During his PhD studies, he spent two months as a visiting researcher at the Karlsruhe Institute of Technology, Germany (2018) and a month at the University of Oxford, United Kingdom (2020). He also published research works in collaboration with researchers from the University of Toronto, Canada. Besides scientific work, he has also been a teaching assistant at several undergraduate and graduate courses at FER and the Croatian Defense Academy dr. Franjo Tuđman. He is a graduate student member of IEEE.

His main research interests within the area of robotics and autonomous systems revolve around motion planning for high-dimensional robots in complex and dynamic environments. He is an author or a co-author of 6 papers published in peer-reviewed journals and 9 papers presented at international conferences. The full list of papers published at the time of writing of the doctoral thesis is given below.

FULL LIST OF PUBLICATIONS

JOURNAL PUBLICATIONS:

1. L. Petrović, J. Peršić, M. Seder and I. Marković. Cross-entropy based stochastic optimization of robot trajectories using heteroscedastic continuous-time Gaussian processes. *Robotics and Autonomous Systems*, 133:103618, 2020, IF: 2.825 (Q2).

2. J. Peršić, L. Petrović, I. Marković and I. Petrović. Online multi-sensor calibration based on moving object tracking. *Advanced Robotics*, 35(3-4):130-140, 2021, IF: 1.247 (Q4).

3. J. Peršić, L. Petrović, I. Marković and I. Petrović. Spatiotemporal Multisensor Calibration via Gaussian Processes Moving Target Tracking. *IEEE Transactions on Robotics*, 37(5):1401-1415, 2021, IF: 6.123 (Q1).

4. F. Marić, L. Petrović, M. Guberina, J. Kelly and I. Petrović. A Riemannian Metric for Geometry-Aware Singularity Avoidance by Articulated Robots. *Robotics and Autonomous Systems*, 145:103865, 2021, IF: 3.120 (Q2).

5. L. Petrović, I. Marković and I. Petrović. Mixtures of Gaussian Processes for Robot Motion Planning Using Stochastic Trajectory Optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Early Access, 2022, IF: 13.451 (Q1).

CONFERENCE PUBLICATIONS:

1. L. Petrović, Š. Ileš and J. Matuško. Self-learning Model Predictive Control Based on the Sequence of Controllable Sets. *International Conference on Electrical Drives and Power Electronics (EDPE)*. Dubrovnik, Croatia, 359–364, 2017.

2. L. Petrović, I. Marković and M. Seder. Multi-agent Gaussian Process Motion Planning via Probabilistic Inference. *IFAC Symposium on Robot Control (SYROCO)*. Budapest, Hungary, 160–165, 2018.

3. M. Seder, L. Petrović, J. Peršić, G. Popović, T. Petković, A. Šelek, B. Bićanić, I. Cvišić, D. Josić, I. Marković, I. Petrović and A. Muhammad. Open Platform Based Mobile Robot Control for Automation in Manufacturing Logistics. *IFAC Workshop on Robot Control (WROCO)*. Daejeon, South Korea, 160–165, 2019.

4. F. Marić, O. Limoyo, L. Petrović, T. Ablett, I. Petrović and J. Kelly. Fast Manipulability Maximization Using Continuous-Time Trajectory Optimization. *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS)*. Macau, China, 8258–8264, 2019.

5. L. Petrović, J. Peršić, M. Seder and I. Marković. Stochastic optimization for trajectory planning with heteroscedastic Gaussian processes. *European Conference on Mobile Robots (ECMR)*. Prague, Czech Republic, 1–6, 2019.

6. L. Petrović, F. Marić, I. Marković and I. Petrović. Gaussian Processes Incremental Inference for Mobile Robots Dynamic Planning. *IFAC World Congress (IFAC-WC)*. Berlin, Germany, 9584–9589, 2020.

7. T. Petković, L. Petrović, I. Marković and I. Petrović. Ensemble of LSTMs and feature selection for human action prediction. *International Conference on Intelligent Autonomous Systems (IAS)*. Singapore, Singapore, 429–441, 2021.

8. E. Szymańska, L. Petrović, I. Marković and I. Petrović. Mobile Robot Teleoperation via Android Mobile Device with UDP Communication. *International Convention on Information, Communication and Electronic Technology (MIPRO)*. Opatija, Croatia, 1143–1148, 2021.

9. L. Petrović, F. Marić, I. Marković, J. Kelly and I. Petrović. Trajectory Optimization with Geometry-Aware Singularity Avoidance for Robot Motion Planning. *International Conference on Control, Automation and Systems (ICCAS)*. Jeju, South Korea, 1760–1765, 2022.

ŽIVOTOPIS

Luka Petrović rođen je 1995. godine u Zagrebu. Zvanja prvostupnika i magistra elektrotehnike i informacijske tehnologije stekao je na Fakultetu Elektrotehnike i Računarstva Sveučilišta u Zagrebu 2015. te 2017. godine Za izvrstan akademski uspjeh tijekom diplomskoga studija na FER-u nagrađen je dekanovom nagradom "Josip Lončar" 2016. godine te brončanom plaketom "Josip Lončar" 2017. godine. Također je nagrađen rektorovom nagradom za izvrstan studentski istraživački rad naslova "Decentralizirano upravljanje višeagentskim robotskim sustavom" te stipendijom za izvrsnost Sveučilišta u Zagrebu.

Nakon završetka studija 2017. godine zapošljava se na Zavodu za automatiku i računalno inženjerstvo na FER-u kao istraživač na projektu "DATACROSS – Napredne metode i tehnologije u znanosti o podacima i kooperativnim sustavima", financiranom iz fondova Europske unije (EU). Iste godine započinje doktorski studij elektrotehnike na FER-u. Aktivno je sudjelovao kao istraživač na projektu "L4MS – Logistics for Manufacturing SMEs", financiranom iz EU fondova (2018-2019). Tijekom doktorskoga studija proveo je dva mjeseca kao gostujući istraživač na Karlsruhe institutu za tehnologiju u Njemačkoj (2018.) te mjesec dana na Sveučilištu u Oxfordu u Ujedinjenom Kraljevstvu (2020.). Također je surađivao i publicirao radove s istraživačima sa Sveučilišta u Torontu u Kanadi. Osim znanstvenoga rada, sudjelovao je i kao asistent u nastavi na nekoliko predmeta na preddiplomskom i diplomskome studiju na FER-u i Hrvatskom vojnom učilištu dr. Franjo Tuđman. Član je udruge IEEE.

Njegovi glavni istraživački interesi u području robotike i autonomnih sustava nalaze se u području planiranja gibanja za visokodimenzijske robote u složenim i dinamičkim okruženjima. Autor je ili suautor 6 znanstvenih radova u časopisima s međunarodnom recenzijom te 9 radova prezentiranih na međunarodnim konferencijama.

COLOPHON

This document was typeset and inspired by the typographical look-and-feel `classicthesis` developed by André Miede, which was based on Robert Bringhurst's book on typography *The Elements of Typographic Style*, and by the `FERElemental` developed by Ivan Marković whose design was based on `FERBook` developed by Jadranko Matuško.