Sampling-based Motion Planning in Configuration and State Spaces
Using supervised learning tools

Bharatheesha, Mukunda

DOI
10.4233/uuid:dd56840f-050e-419c-9ceb-8eca3be414bd

Publication date
2018

Citation (APA)
Bharatheesha, M. (2018). *Sampling-based Motion Planning in Configuration and State Spaces: Using supervised learning tools*. [Dissertation (TU Delft), Delft University of Technology].
https://doi.org/10.4233/uuid:dd56840f-050e-419c-9ceb-8eca3be414bd

SAMPLING-BASED MOTION PLANNING IN CONFIGURATION AND STATE SPACES

USING SUPERVISED LEARNING TOOLS

SAMPLING-BASED MOTION PLANNING IN CONFIGURATION AND STATE SPACES

USING SUPERVISED LEARNING TOOLS

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T. H. J. J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 4 juli 2018 om 12:30 uur

door

Mukunda BHARATHEESHA

Master of Science in Embedded Systems,
Technische Universiteit Delft, Nederland
geboren te Bangalore, India.

Dit proefschrift is goedgekeurd door de promotor:

Prof. dr. ir. M. Wisse

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof. dr. ir. M. Wisse	Technische Universiteit Delft, promotor

Onafhankelijke leden:

Prof. dr. ir. C. Witteveen	Technische Universiteit Delft
Dr. J-P. Laumond	LAAS-CNRS, France
Prof. dr. J. de Schutter	Katholieke Universiteit Leuven, Belgium
Prof. dr. P. G. Plöger	Hochschüle Bonn-Rhein-Sieg, Germany
Prof. dr. ir. R. R. Negenborn	Technische Universiteit Delft
Dr. J. Kober	Technische Universiteit Delft

The research presented in this thesis has received financial support from the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 609206.

The author expresses his sincere thanks to Ir. Wouter J. Wolfslag for his support and collaborations in realizing some of the results in the second part of this thesis.

Keywords: sampling-based motion planning, supervised learning, kinodynamic planning, distance metric approximation

Printed by: Ipskamp Drukkers B. V., Enschede.

Front & Back: Cover design by Maryam Sharify, TU Delft.

Author e-mail: mukunda1028@gmail.com

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

To the memory of my dear uncle Gunda.

SUMMARY

Robotic systems are the workhorses in practically all automated applications. Manufacturing industries, warehouses, elderly care, disaster rescue and (unfortunately) warfare are example applications where human life has benefited from robotics. By precisely *planning* and *controlling* their motions via computer programs, real world tasks can be performed with high levels of accuracy and repeatability. Devising methods and algorithms that generate such motions by a) correctly reporting and finding the desired motion if it exists and b) doing so as fast as possible, has constituted the field of robot motion planning research over the last four decades.

In recent years, the Industry 4.0 initiative has provided a promising avenue for further advances in industrial automation. Modular, quickly reconfigurable and versatile robotic systems that safely collaborate with humans hold the key to future industrial automation. This is a challenging endeavor from an industrial and an academic perspective and inspires the work in this thesis. In alignment with these perspectives, this thesis is presented in two parts.

In the first part, we propose methods and frameworks to effectively utilize open source implementations of configuration space planners to realize flexible and robust solutions for bin picking. To this end, we make three contributions achieved via collaborations with multiple research groups.

First, we present a framework using model-based algorithm configuration tools to make the best choice of a motion planning algorithm for a given application. This framework abstracts the complex interaction between algorithm implementation parameters and the algorithm performance as a *blackbox* tuning process. The benefits of such a tuning process is shown on three different motion planning applications with industrial robots. From the perspective of an end-user looking to utilize an open source motion planning software, this framework serves a key purpose. The user can solely focus on specifying performance indicators such as planning times, path lengths and so forth and eventually make an informed choice of a planning algorithm.

As a second contribution, we present a motion planning module as a part of an integrated software framework used in the robotic system that won the Amazon Picking Challenge in 2016. The key message here is that open source motion planning frameworks are effective tools to model and optimize a robotic system design for a given application. Without having to invest in actual hardware, some important initial design questions can be answered. For example, which robot manipulator is best suited for a desired application? How can flexibility be incorporated in the system design to enhance reusability? This is an interesting prospect for Small and Medium Scale Enterprises (SMEs) looking at economical automation solutions.

As the third contribution, we present a reactive collision avoidance framework in a collaborative bin picking scenario. The main purpose of this contribution is to answer the question: What is the main challenge from a motion planning perspective in realizing *plug*

and work robotic solutions for SMEs? It is observed that the main challenge arises from two opposing requirements in achieving a reliable and robust reactive behavior. On the one hand, a plug and work robot hardware has to be easily installable and reconfigurable with preferably low costs. This points towards setups with small-sized robotic manipulators. On the other hand, reactive motion generation principles rely on availability of redundant information in a robotic system to obtain reactive motions. This points towards increased robot complexity, size and costs. Utilizing shared efforts across multiple research organizations within Europe, a prototype robotic system that is easily installable and reconfigurable has been realized.

A fundamental limitation of planning in the configuration space is that it is impossible to realize versatile motions where a precise end velocity is desired. This limitation can be addressed by planning in the state space which also implicitly accounts for the physical laws governing the motion of a robot. However, sampling-based planning in state space (also called kinodynamic planning) is computationally intensive and challenging to realize in practice. This inspires the second part of the thesis which focuses on development of methods to speed up sampling-based motion planning in state space.

In the second part of the thesis, the goal is to answer the question: Is it possible to achieve planning speeds in state space that are comparable to planning speeds in the configuration space? We pursue this goal by considering the Rapidly exploring Random Tree (RRT) planner in state space. We make two contributions that significantly alleviate the computation times of the following steps in an RRT: (i) choosing the state in the tree that is nearest to the randomly sampled state (ii) steering from the nearest state to the randomly sampled state.

In the first contribution, we present a framework to approximate the *distance (pseudo) metric* in state space using supervised learning tools. It is observed that reliable approximations are possible with speeds that are up to 3 orders of magnitude faster relative to explicitly solving for the distance metric. It is also observed that, the main limitation of this contribution is that the optimal control formulation used to compute the distance metric does not have a parameterization of the steering inputs that can be learned. This limitation inspires the following contribution.

The key idea of the second contribution is the utilization of indirect optimal control principles that result in a steering input formulation as a (non-linear) function of generalizable parameters called *co-states*. This enables the extension of the supervised learning idea to also approximate these parameters and hence the steering input. It is observed that learning the steering inputs further speeds up the planning times by an order of magnitude relative to the earlier results. It is also shown that with assumptions such as small time reachability and well bounded costs, the proposed learning-based RRT approach is probabilistically complete.

The 2-dimensional state space of a simple pendulum is considered for the proof of concept for the above contributions. A planning time of $\sim$2.4 s is achieved to plan a *swingup* motion for the pendulum. This seems quite slow compared to the planning speeds of configuration space planners that generate plans within a tenth of a second in $7-8$ dimensional planning spaces. However, the achieved results are faster than current state of the art solutions for motion planning in state spaces with similar dimensionality. Thus, reaching planning times equivalent to or better than what is achievable in configuration space still remains an elusive goal. Nevertheless, the achieved results serve as encouraging signs to pursue further research in this direction.

SAMENVATTING

Robots zijn de werkpaarden in praktisch alle geautomatiseerde systemen. De maakindustrie, distributiecentra, ouderenzorg, rampenbestrijding en (helaas) oorlogsvoering zijn voorbeelden van toepassingen waar mensen baat hebben bij robotica. Door het nauwkeurig *plannen en regelen* van hun bewegingen middels software, kunnen taken in de echte wereld worden uitgevoerd met grote precisie en herhaalbaarheid. Het bedenken van methodes en algoritmes die zulke bewegingen genereren door a) het correct vinden en doorgeven van de gewenste beweging, mits die bestaat, en b) deze zo snel mogelijk uitvoeren, heeft in 40 jaar het veld van robot "motion planning" gevormd.

In de laatste paar jaren heeft het "Industry 4.0" initiatief een veelbelovende weg ingeslagen voor vooruitgang in de industriële automatisering. Modulaire, veelzijdige en snel te herconfigureren robotische systemen, die veilig samenwerken met mensen, spelen een sleutelrol in de toekomst van industriële automatisering. Dit is een uitdagende onderneming vanuit zowel een industrieel als academisch perspectief, en het heeft het werk in deze thesis geïnspireerd. In overeenkomst met deze perspectieven, bestaat deze thesis uit twee delen.

In het eerste deel stellen we methodes en frameworks voor die effectief gebruik maken van "open source" implementaties van "configuration space planners", om flexibele en robuuste oplossingen voor "bin picking" te realiseren. Hiertoe hebben we drie bijdragen geleverd, via samenwerkingen met meerdere onderzoeksgroepen.

Allereerst presenteren we een framework dat gebruikt maakt van "model-based" algoritme configuratie, om voor een bepaalde toepassing de beste keuze voor een motion planning algoritme te maken. Dit framework abstraheert de complexe interactie tussen de parameters voor implementatie van het algoritme en de bijbehorende prestaties als een "blackbox tuning" proces. De voordelen van een dergelijk proces zijn gedemonstreerd op drie verschillende toepassingen van motion planning voor industriële robots. Het gepresenteerde framework speelt een sleutelrol voor een eindgebruiker die een open source motion planner wil gaan gebruiken. De gebruiker hoeft zich alleen te richten op het specificeren van prestatie-indicatoren zoals planningstijden, weglengtes, enzovoort, om uiteindelijk een geïnformeerd besluit te kunnen nemen over welk planning algoritme te gebruiken.

Als tweede bijdrage presenteren we een motion planning module, als deel van een geïntegreerd software framework, dat gebruikt is in het robotische systeem dat in 2016 de "Amazon Picking Challenge" heeft gewonnen. De boodschap hiervan is dat open source motion planning frameworks effectieve instrumenten zijn voor het modelleren en optimaliseren van het ontwerp van een robotisch systeem voor een gegeven toepassing. Een aantal belangrijke ontwerpvragen kunnen beantwoord worden, zonder te hoeven investeren in daadwerkelijke hardware. Bijvoorbeeld: wat voor robotarm is het meest geschikt voor de gewenste toepassing? Hoe kan flexibiliteit in het systeem gebruikt worden om hergebruik te bevorderen? Dit is een interessant vooruitzicht voor Midden- en Klein Bedrijven (MKBs) die op zoek zijn naar economisch haalbare automatiseringsoplossingen.

Als derde bijdrage presenteren we een reactief botsingpreventie framework in een collaboratief bin picking scenario. Het hoofddoel van deze bijdrage is het beantwoorden van de volgende vraag: Wat is de grootste uitdaging in het realiseren van “plug-and-work” oplossingen voor MKBs, vanuit een motion planning perspectief? We hebben gezien dat de grootste uitdaging voortkomt uit twee tegengestelde eisen voor het halen van betrouwbaar en robuust reactief gedrag. Aan de ene kant moet een plug-and-work robot gemakkelijk, en bij voorkeur goedkoop, te installeren en opnieuw te configureren zijn. Dit stuurt aan op opstellingen met kleine robots. Aan de andere kant vertrouwen reactieve motion planning principes op de beschikbaarheid van overvloedige informatie vanuit het robotische systeem, om reactieve bewegingen te kunnen genereren. Dit stuurt aan op complexere, grotere en duurdere robots. Middels gedeelde inspanning van meerdere onderzoeksinstellingen in Europa, is een prototype van een robotisch systeem gerealiseerd dat gemakkelijk te installeren en opnieuw te configureren is.

Een fundamentele beperking van plannen in *configuration space* is dat het onmogelijk is om veelzijdige bewegingen te maken, terwijl een nauwkeurige tipsnelheid gewenst is. Deze beperking kan worden aangepakt door te plannen in *state space*, waarin impliciet rekening wordt gehouden met natuurwetten die de beweging van een robot bepalen. Echter, sampling-gebaseerd plannen in state space (ook wel *kinodynamic* plannen genoemd) vereist veel rekenkracht en is uitdagend om in de praktijk te realiseren. Dit heeft het tweede deel van deze thesis geïnspireerd, waarin wordt gefocust op ontwikkeling van methodes om sampling-gebaseerde motion planning in state space sneller te maken.

In het tweede deel van deze thesis is het doel het beantwoorden van de vraag: Is het mogelijk om motion planning in state space even snel te maken als planning in configuration space? We streven ernaar dit doel te bereiken via de *Rapidly exploring Random Tree (RRT)* planner in state space. We hebben twee bijdrages die de rekentijd in de twee volgende stappen van RRTs significant verbeteren: (i) het kiezen van de state die het dichtste bij de willekeurig gesampelde state ligt, en (ii) het sturen van deze state naar de willekeurig gesampelde state.

In de eerste bijdrage presenteren we een kader om de *(pseudo)afstandsmaat* in state space te benaderen via “supervised learning”. We zien dat betrouwbare benaderingen mogelijk zijn met rekensnelheden tot drie maal sneller ten opzichte van het expliciet oplossen van de afstandsmaat. Tevens zien we dat de grootste beperking van deze bijdrage is dat de “optimal control” formulering gebruikt voor het berekenen van de afstandsmaat geen parametrering van de sturende inputs heeft die geleerd kan worden. Deze beperking heeft de volgende bijdrage geïnspireerd.

Het belangrijkste idee van de tweede bijdrage is het gebruik van indirecte optimal control principes, die resulteren in een formulering van sturende input in de vorm van een (niet-lineaire) functie of generaliseerbare parameters die we “co-states” noemen. Dit maakt uitbreiding met het supervised-learning idee mogelijk, zodat ook deze parameters, en daarmee de sturende input, benaderd kunnen worden. We zien dat het leren van sturende inputs de planningstijden met een ordegrootte versnelt ten opzichte van de eerdere resultaten. We laten zien dat met aannames zoals “small time reachability” en “well bounded costs”, de voorgestelde “learning-based RRT” benadering probabilistisch compleet is.

De 2-dimensionale state space van een simpele slinger wordt gebruikt voor het proof of concept van de bovenstaande bijdrages. Een planningstijd van ~2.4 s is gehaald voor het plannen van een opzwaaiende beweging van de slinger. Dit lijkt langzaam in vergelijking met de planningssnelheid van configuration space planners, die binnen een tiende seconde plannen genereren in een $7-8$ dimensionale planningsruimte. Echter, de behaalde resultaten zijn sneller dan de huidige state-of-the-art oplossingen voor motion planning in state space met vergelijkbare dimensionaliteit. Dus, het halen van planningstijden gelijk aan of beter dan wat te halen is in configuration space blijft een moeilijk te bereiken doel. Desalniettemin, de behaalde resultaten moedigen aan onderzoek in deze richting te blijven nastreven.

CONTENTS

1

INTRODUCTION

1

MOTION PLANNING has been an integral part of robotics in realizing autonomous behaviors in unstructured and unknown environments. Enabling a robotic system to autonomously plan feasible motions from a given starting configuration to a desired goal configuration is the subject matter of motion planning. Providing strong or weak notions of guarantees of finding such a plan and obtaining a plan as quickly as possible are two of the fundamental challenges in motion planning. These challenges have inspired a large volume of motion planning research since the early 1970s. The initial works focused primarily on the first challenge and the last two decades have seen a focus shift towards addressing the second challenge. A foundational concept that is common to practically all methods that address these challenges is the idea of planning spaces.

1.1. PLANNING SPACES

A key component in the development of motion planning algorithms is the transformation of a real world motion planning problem into a representation understandable by a computer. This transformation involves identifying a set of independent variables that represent a robot in the context of its working environment. The orthogonal vector space spanned by these independent variables is called a *planning space*. There are two commonly used planning spaces namely, the configuration space and the state space. The configuration space is defined by a set of generalized coordinates encoding positional information of a robot in its environment. Similarly, the state space is defined by the set of generalized coordinates that encode both positions and velocities of a robot in its environment.

Typically the configuration space is represented with $\mathscr{C}$ and the state space is represented with $\mathscr{X}$. The planning space consisting of robot configurations or states that do not collide with obstacles in the environment is represented with $\mathscr{C}_{\text{free}} \subseteq \mathscr{C}$ and $\mathscr{X}_{\text{free}} \subseteq \mathscr{X}$ respectively. Similarly, configurations or states that involve a collision with an obstacle are represented with $\mathscr{C}_{\text{obs}} \subseteq \mathscr{C}$ and $\mathscr{X}_{\text{obs}} \subseteq \mathscr{X}$. In the state space, an additional representation is considered, $\mathscr{X}_{\text{ric}} \subseteq \mathscr{X}$. $\mathscr{X}_{\text{ric}}$ represents the region of inevitable collisions where the state of a robot is such that a collision with an obstacle is unavoidable.

The concept of a planning space enables the representation of a robot as a point in a higher dimensional space. Subsequently, a *graph structure* can be created in the planning space with the nodes of the graph being robot configurations or states and the edges of the graph being feasible paths or trajectories taken by the robot. This idea, first introduced in the seminal work of [86], forms a crucial entry point to the research and development of motion planning algorithms to the present day. Graph structure representation of the motion planning problem creates an infrastructure to use a computer to query for a solution using graph search methods such as A*[48] and Dijkstra's algorithm [32].

1.2. COMBINATORIAL VS SAMPLING-BASED PLANNING

Transforming a motion planning problem to a graph structure representation in the planning space is certainly desirable. In fact, the creation of the graph structure is one of the fundamental subjects of motion planning research. The process of how the graph structure is constructed forms the basis for broadly classifying motion planning algorithms in two categories namely, Combinatorial methods and Sampling-based methods.

Typically $\mathscr{C}$ or $\mathscr{X}$ are continuous spaces as they represent the real world. However, an appropriate discretization is required to construct the graph structure. Combinatorial methods discretize the planning space without loss of information about the real world problem that is being represented. Such discretizations are called *exact* representations of a real world motion planning problem and consequently these methods exclusively address the challenge of providing strong guarantees on finding a motion planning solution to a given problem, if one exists. In other words, the existence or non-existence of a solution to a given motion planning problem will always be correctly reported by combinatorial methods. In motion planning literature [119, 73, 78], such guarantees are called completeness guarantees. Elegant and optimal solutions to classes of motion planning problems involving robots that translate in a 2D environment with polygonal obstacles have been realized with combinatorial methods including upper bounds on planning times [119]. However, constructing exact representations of planning spaces, particularly $\mathscr{C}_{\text{obs}}$ and/or $\mathscr{X}_{\text{obs}}$ is a complex problem [78].

Consider for example a pick and place problem for a 7-degrees of freedom industrial manipulator, fixed to a platform on the ground, operating in a 3D space with static rigid obstacles. It is perhaps possible (with an adequately fine discretization) to construct an exact representation of $\mathscr{C}_{\text{free}}$ or $\mathscr{X}_{\text{free}}$. With computers getting ever faster, performing a sub-optimal search in a 7-dimensional search space could also be realistic. However, constructing an exact representation of $\mathscr{C}_{\text{obs}}$ and/or $\mathscr{X}_{\text{obs}}$ is impractical due to the fundamental limitation imposed by the underlying one-to-many mapping between a real-world obstacle and the obstacle representation in a planning space. While exact representations are idealistic, one might accept a relaxed form of this requirement to work with convex approximations of obstacle topologies. This has also been shown to be a difficult problem to solve within practical time bounds (see Section 4.3 in [78]). It is important to note that exact representations of $\mathscr{C}$ or $\mathscr{X}$ is a necessary pre-requisite to use combinatorial methods. Recollecting the two fundamental challenges of motion planning, it is evident from the above that combinatorial methods strive to address the first challenge of providing strong guarantees on finding a motion planning solution, if one exists. However, this comes at the cost of synthesizing complex representations of obstacle topologies and also planning times that are practically unacceptable for planning problems in high dimensional planning spaces. While combinatorial methods certainly open the gateways toward algorithmic pursuits in a theoretical sense, they are not promising directions to pursue when viewed from the perspective of deploying autonomous robotic systems in the real world as highlighted in [73, 78].

What if non-exact representations of planning spaces and the consequent weaker notions of completeness are admitted and more focus is laid on the challenge of planning time? What if the complexity of constructing exact representations of $\mathscr{C}_{\text{obs}}$ and/or $\mathscr{X}_{\text{obs}}$ can be worked around using a collision checking module that tests if a given robot configuration or state is in collision with an obstacle using forward kinematics? These questions have inspired an alternative, but an extremely effective field of motion planning algorithms called Sampling-based methods and they provide the following benefits over combinatorial methods:

- Exact representations of the configuration space $\mathscr{C}$ or the state space $\mathscr{X}$ (and the associated subspaces) is not required.
- Discretization is achieved by *sampling* directly from $\mathscr{C}_{\text{free}}$ and $\mathscr{X}_{\text{free}}$.

- The problem of obstacle space representation is bypassed with the use of a collision checking module consequently opening up realistic possibilities of planning in high dimensional spaces.

The primary benefit of sampling-based methods is that exact planning space representations are not used. Typically, exact representations [18, 59, 73] are constructed based on the geometry of the robot and the environment it is operating in. As a result, changing geometries such as re-organization of robot work cells or moving obstacles require a reconstruction of the exact representations. In higher dimensional planning spaces, this can be a tedious process. As sampling-based methods do not require such representations, they can adapt to changing geometries and thus provide a great deal of flexibility. The second benefit comes from the indirect manner of discretization of the planning space achieved via random [65, 79, 72], deterministic or informed sampling [39, 58] of the planning space. This allows one to address different qualitative goals such as exploration of a planning space, greedy strategies to achieve a fast but sub-optimal solution or a weighted choice between exploration and greedy strategies [130, 147]. The final benefit comes from the use of a stand alone collision checking module that bypasses the complexity of constructing obstacle representations, particularly when working with high dimensional planning spaces where unique mappings between obstacles and robot configurations or states do not exist. This enables one to explicitly focus on planning in $\mathscr{C}_{\text{free}}$ and $\mathscr{X}_{\text{free}}$ and eventually use the collision checking module as a form of *quality control* to ensure feasibility of solutions. This benefit is further exploited to defer the collision checking process until absolutely necessary [15, 49].

1.3. RANDOMIZED SAMPLING-BASED PLANNING

The virtues of sampling-based methods seem to outweigh the need for absolute completeness guarantees. This is evidenced by the massive advances of literature in the past two decades that build on one of the two fundamental sampling-based methods namely, Probabilistic Road Maps (PRM) [65] and Rapidly exploring Random Trees (RRT) [79]. Both methods build a graph structure by (uniform) random sampling of the planning space with the difference being the type of the graph structure that results from each approach. The PRM approach constructs a *roadmap* which is a graph structure that can have cycles. In other words, there could potentially be multiple paths from a given start and goal nodes in a roadmap. In the RRT approach, a *tree* graph structure is built where no cycles are allowed and thus a start and a goal node in a tree can be connected by one path. A key property common to both PRM and RRT approaches is the probability of finding such a path if one exists approaches 1 as the number of samples approaches infinity. This is called *probabilistic completeness*. Compared to the absolute completeness guarantees that are provided by combinatorial methods, probabilistic completeness is a weaker notion of guarantee that a motion planning solution will be found for a given problem. However, various works in literature (see Section 1.4.2) have shown that the notion of probabilistic completeness is sufficient for several practical problems in motion planning.

From an algorithmic perspective, the PRM approach is composed of two stages namely a construction phase and a query phase. As the names imply, the construction phase is the process of building the graph structure and the query phase is the process of finding a path

through the graph. The two step approach enables the possibility to query for solutions for multiple start-goal pairs after a single construction phase. In [131], the authors also provide a general condition to gracefully terminate the construction phase depending on the achieved coverage of the free configuration space. However, a common drawback of PRM-based methods is that the construction phase has to be repeated if the working environment of the robot changes. The RRT approach however, constructs the graph structure while actively processing a query for a path and the graph structure growth typically stops after the goal has been reached. Accordingly, PRM (based) methods are classified under *multi query* planning and RRT (based) methods under *single query* planning in motion planning literature. The term *node* is commonly used in planning literature to refer to a configuration or state within a graph structure. The research in this thesis is centered around the RRT approach as it offers the potential to also work with changing environments. Furthermore, the solutions and methods devised in this thesis are directly extendable to the PRM based approaches. Therefore, in the rest of this thesis, the focus will solely be on the RRT (based) approaches.

1.3.1. RAPIDLY EXPLORING RANDOM TREE (RRT)

The concept of a Rapidly exploring Random Tree (RRT) was proposed by Steven Lavalle as a new tool for path planning in [79]. Given a start-goal pair, the RRT iterates over the following steps to build a tree graph structure after initializing the tree with the starting node (also called sometimes as root node):

I Randomly sample a node from $\mathscr{C}_{\text{free}}$ or $\mathscr{X}_{\text{free}}$.

II Determine the node in the graph that is *nearest* to the random sample based on a chosen distance metric.

III Extend the graph structure by *steering* from the nearest node to the randomly sampled node.

IV Terminate if the goal node is reached or the maximum number of iterations have been achieved.

The resulting graph structure and its consequent influence on finding a solution to a motion planning query is critically defined by steps II and III. These steps together enable the RRT to quickly cover unexplored regions of a planning space while attempting only the most likely connections. As noted in [79], the RRT is constructed using a simple set of steps but devising methods that accurately perform these steps is not straight forward. These steps are particularly challenging in the context of motion planning in state space which we elaborate further in Section 1.5.1 and Section 1.5.2.

1.4. MOTIVATION - PART I

The different benefits offered by sampling-based methods combined with the generic representation infrastructure they are based on makes them promising candidates to study further in the pursuit of realizing and deploying autonomous robotic systems in the real world. A robotic system in the real world is a broad and an abstract concept. In this thesis, we focus on sampling-based motion planning for fixed serial link manipulators in industrial and academic

applications. Particularly, we study two related, yet contrasting aspects in the field of motion planning for manipulators namely, sampling-based motion planning in configuration and state spaces.

The research conducted in this thesis is presented in two parts: *industrial applications* with sampling-based motion planning in the *configuration space* and *academic applications* with sampling-based motion planning in the *state space*. The first part of this thesis is motivated by the quest for realizing *plug and work* robots for Small and Medium Scale Enterprises (SMEs) as a part of the EU project Factory-in-a-Day [145]: focused on developing and deploying reusable open source software components. The second part of this thesis is motivated by the quest for speeding up planning times of sampling-based motion planners in state space: focused on development of new methods using supervised learning as a tool to enable the generation of fast and dynamically feasible motion planning solutions. These topics will be further elaborated in the following sections.

1.4.1. SAMPLING-BASED PLANNING IN CONFIGURATION SPACE

Sampling-based motion planning in the configuration space of manipulators has long been an active topic of research. In addition to the research, recent efforts from the open source robotics community[1] have enabled these methods to take shape as off-the-shelf software components. These components could prove to be attractive tools for Small and Medium Scale Enterprises (SMEs) focused on developing robotic solutions where a standard production line with one or multiple robots in a cage is impractical from an economic perspective. This motivates the work in the first part of this thesis, the industrial application of bin picking is studied with configuration space planning as an important functional component.

1.4.2. RELEVANT LITERATURE

Development of different configuration space planners based on the RRT approach has seen a steady progress since the basic RRT [79] was first proposed in the late 90s. Perhaps the simplest yet the most significant improvement of the basic RRT , the RRT-Connect [72] was subsequently proposed where two trees were seeded, one at the start and the other at the goal node that grew towards each other. This variant lead to an exponential reduction in planning time and remains as one of the most commonly used configuration space planner in many practical applications to date. Subsequent development of RRT-based approaches predominantly focused on leveraging different heuristics to *guide* the growth of the tree in desired directions of $\mathscr{C}_{\text{free}}$ depending on the problem being solved [141]. One of the significant developments in this direction, the Transition-based RRT (T-RRT) proposed the idea of combining costmap heuristics [69] to guide RRT exploration only along directions that would eventually lead to low cost paths. Yet another idea that received quite a significant research attention was the enhancement of the sampling-process by augmenting robot or environment specific information [103, 40]. Parallel to the development of these variants, an important line of research that focused on reasoning about the optimality of the generated paths based on a certain measure, was pioneered by the work of Karaman and Frazzoli [62]. The associated algorithm, called RRT* and developments that build on RRT* [40, 58] are commonly referred as asymptotically optimal planners which guarantee the optimality of the

[1] www.ros.org, www.osrfoundation.org, www.rosindustrial.org

solution returned for a given instance of the tree. They further guarantee that the returned solution will be globally optimal as the number of samples tends to infinity. For further details and a comprehensive overview of sampling-based motion planning methods, the reader is referred to the work by Tsianos et. al [139].

Given the immediate practical relevance of RRT (and PRM) variants to robotic applications, they took shape as open source software components with the seminal developments in [135, 133]. Open source software has also been complemented with notable commercial implementations of motion planning algorithms such as the Kineo-CAM planning tools [74]. While such off-the-shelf software components are certainly beneficial to the robotics community, their complete potential is realized when they are used in the context of integrated robotic applications such as bin picking, autonomous navigation and so on. The solutions generated by the class of above mentioned algorithms are categorized as *global* solutions which are typically composed of short path segments (edges of the graph structure). However, global solutions alone are insufficient when practical robotic applications are considered. Especially, if the environment around the robot is changing (due to moving obstacles or people), the feasibility of a motion plan needs to be frequently evaluated over the various path segments as some segments might become invalid due to collisions. Motion planning methods that address this problem are categorized as *local* planning methods.

Local planners focus on the process of modifying or replacing the smaller segments of a global path to ensure the resulting path is feasible (typically collision free). Typically, some form of environment sensing such as a camera or a proximity sensor mounted on the robot is used in local planners. One of the first contributions in this area, namely Artificial Potential Fields (APF) [66] was proposed with the fundamental idea of *repelling* the robot from obstacles and *attracting* the robot towards the goal. While the native method in [66] had the limitation of getting stuck in local minima caused by the nullification of equal attractive and repulsive effects, the idea opened up the avenue for realizing *reactive* robot behavior. For example, Barraquand and Latombe [7] proposed the idea of using randomization to escape local minima. Their work also proposed a method to extend these principles also to the global planning problem. The APF-based approaches provided an effective framework for realizing collision avoidance behaviors using sensory information. However, they were also limited by the fact that, the framework was suited for realizing only one qualitative behavior. This limitation inspired the development and extension of the operational space task function formalism [66, 125] with multi-objective optimization tools to incorporate multiple qualitative requirements, such as maintaining a certain pose or posture of the robot while maintaining a certain distance from obstacles. The central idea being the inclusion of multiple *tasks* (with an optional prioritization) in accordance with the desired qualitative requirements and eventually solving the local motion segment (re) generation problem as an optimization problem. This idea has received a wide level of acceptance in the humanoid and quadrupedal robotics community as *whole body control* and the work in [76] provides an overview of the associated fundamental challenges. Such systems typically require several tasks to be performed at the same time to maintain a stable posture [128, 116]. Efforts from different research groups have also enabled the creation of off-the-shelf software components such as the Stack of Tasks (SOT) framework [88] and the eTaSL/eTC task specification language and controller framework [2].

While these methods have not been actively used with traditional industrial manipulators, they certainly have created the pathway for specification and realization of multiple qualitative tasks involving industrial manipulators. This is particularly interesting in the current efforts towards realizing applications where a collaborative behavior is desired between a human being and a robotic manipulator. For example, it is pertinent that a certain pick or a place task is accomplished, while accounting for motion speed reduction if a human is detected in the vicinity or a modification of the path if a previously unaccounted obstacle is blocking the robot path.

1.4.3. PROBLEM STATEMENT - PART I

A primary benefit of an off-the-shelf software component that provides motion planning functionality (or any other relevant functionality such as computer vision, grasp synthesis and so on) is the ability to build *flexible* robotic solutions for bin picking applications in SMEs. This would enable SMEs to deal with seasonal products in a working environment that is frequently modified depending on the product. Additionally, it is common to have humans in the vicinity of the robot and even working collaboratively. In such a scenario, a *one (robotic) solution fits all* approach will become difficult to synthesize and with strict time constraints (for example, a supermarket promoting a pack of noodles with *spring flavours* or a *new formula* cold cream for the winter), there is limited (re) installation time for adapting a current robotic solution to a new product.

Flexibility in robotic solutions is an attractive benefit. However, do the current infrastructures support the development of software modules that can be quickly ported across different applications? Can these off-the-shelf components function reliably and robustly as demanded by industrial applications? In this context, the first part of this thesis will study the use of open source software components for configuration space planning. In particular we focus on the following practical questions: Given an off-the-shelf configuration space planning software infrastructure and the associated dependencies,

1. How can we choose what is the best configuration space planner for a given bin picking scenario? (Chapter 2)

2. What are the important challenges in integrating motion planning software components with other relevant components to realize a reliably functioning bin picking application? (Chapters 3 and 4)

Along the journey to answer these questions, two important milestones have been reached. The first being the development of a motion planning module for the world championship winning robotic system for the Amazon Picking Challenge 2016. The second being the realization of a motion module with reactive behavior for a collaborative robotic system for bin picking.

1.5. MOTIVATION - PART II

The configuration space of robotic manipulators serves as an attractive abstraction for the creation of the graph structures to solve motion planning problems. Well defined metrics such as the Euclidean distance, Manhattan distance and Mahalanobis distance are available to accurately quantify the relation between different nodes of the graph structure. The

existence of such metrics is a critical factor in reasoning about the quality of the solutions generated by motion planning algorithms. For example, using the Euclidean distance as a metric gives a reliable indication of how much the joints of a manipulator must move relative to their previous positions and a summation of all such movements over the entire path is a reasonable estimate of the total *distance* from start to goal in the configuration space. Creation of configuration space paths alone is an inadequate specification for path execution on a robot manipulator. A path in the configuration space is a sequence of way points (robot configurations) that only encode spatial information between the desired start and goal configurations. Timing information is later appended via a process called *time parameterization* where velocities and accelerations are computed (using cubic or quintic splines [113] for instance) between way points resulting in a trajectory that can be commanded to the robot's motors for execution.

However, as highlighted in [21], all such time parameterizations do not necessarily result in a trajectory that is efficient or feasible to follow for a robotic system. This is because the motion of a robotic system is also governed by the underlying physical laws that express the evolution of motion over time. In motion planning literature, these effects are termed as *differential constraints* on the planning problem [78, 21, 139]. Additionally, the infeasibility could also arise due to actuator limitations that cannot meet the desired velocity and acceleration variations along the resulting trajectory. Therefore extra care has to be taken while parameterizing configuration space paths such that differential constraints and actuator limitations are accounted for. Recently, there has been a rising interest in this area and it is an active topic of research, also popularly called *trajectory generation*. The initial works in this area took inspiration from the works of [14, 129, 106] that propose methods to parameterize an arbitrary path such that the resulting trajectory is time optimal. The authors of [64] propose a method to generate incremental time optimal parameterizations of a path that account for robot dynamics and actuator constraints. In [107, 108], the authors propose the use of dynamically feasible time optimal parameterizations of paths generated by sampling-based planners in configuration space. The key challenge in this class of methods is to ensure admissible parameterizations of all path segments is produced so that the entire trajectory is feasible for execution.

Due to the lack of velocity information in the sampled configurations, the velocity associated with each configuration along a plan is arbitrarily decided based on the underlying time parameterization. This is a fundamental limitation for applications where it is not only required to attain a goal configuration but also do so with a certain desired velocity. Consider for example, the problem of picking up objects using a robot from a moving conveyor belt in an industrial production line. The motion planning problem would involve a goal specification with a certain velocity to account for the moving object at the time of pick up. This could consequently speed up the overall cycle time. Another example could be the task of a robotic goal keeper that is tasked with throwing the ball to players located at different distances. This problem requires a planning solution such that the arm of the goal keeper attains different and precise velocities at the release point of the ball depending on the player that the ball is destined for. Perhaps a similar futuristic example would be a robot arm tasked with the problem of throwing and stacking light weight objects. Sampling such goal specifications is impossible in the configuration space and is thus a fundamental limitation of configuration space planners.

Planning in the state space allows one to address these limitations by providing an elegant and implicit manner to incorporate velocities in the planning space. Also, there is no explicit requirement of time parameterization because the resulting solutions from a motion planning query in state space will be a trajectory encoding both position and velocity information. The resulting solution could thus be potentially transmitted as a sequence of commands to the robot for execution. These aspects inspire the work in the second part of this thesis that is centered around sampling-based motion planning in state space.

1.5.1. SAMPLING-BASED MOTION PLANNING IN STATE SPACE

State space provides an infrastructure to generate motions that can effectively utilize robot dynamics. Typically, robot dynamics are represented with (a set of) differential equations and a solution to these differential equations represents the evolution of state variables over time. In the context of motion planning in state space such solutions could be viewed as potential segments of a global trajectory from a start state to a goal state. A motion planning problem in state space is generally formulated as an optimal control problem where a certain cost is minimized with the system dynamics, the start and the goal states as constraints. As a result, planning in state space opens up avenues to address requirements such as effort minimization, time optimality and so on. Simple cost function formulations would suffice if the focus is purely on factors such as effort or time minimization. Consider a cost function that is quadratic in the torque input to the robot, where the squared value of the applied torque is integrated over the entire trajectory. However, incorporating environmental information about static or dynamic obstacles into the optimization would become increasingly complex [33, 78, 148, 118, 61]. As elaborated in Section 1.2 and Section 1.3, sampling-based motion planning methods provide a framework where generating feasible trajectories and incorporating environmental information can be separately addressed. That is, given a state pair, a dynamically feasible trajectory could be generated with a simpler optimal control problem (as in, without having to account for obstacles while formulating the cost function) and the resulting trajectory can then be checked for collisions with an independent collision checker that also accounts for the environmental information.

The possibility to implicitly incorporate differential constraints to a motion planning problem by choosing to plan in state space presents an interesting set of challenges when viewed from the perspective of sampling-based planning. The ideal distance metric in state space is the optimal cost-to-go between state pairs [20] and thus is a solution of the aforementioned optimal control formulation. Additionally, an optimal control formulation also yields a (locally) optimal trajectory between the start and goal states as a byproduct. However, such formulations are categorized as Two-Point Boundary Valued Problems which are known to be NP-hard (see Section 14.1 in [78], [57]). Furthermore, the authors in [21, 75] also highlight that except for simple 1-D and 2-D systems, it is hard to reason about the existence of exact solutions to such formulations. These aspects critically impact sampling-based methods in state space. This is because of the mechanics of sampling-based methods that incrementally construct a graph representation of the state space as elaborated in Section 1.3. The states of the system constitute the nodes and the trajectories between state pairs constitute the *edges* of the graph structure.

For the purpose of clarity, two of the critical steps from Section 1.3 are repeated here in the context of state space:

- Computing a reliable *distance (pseudo) metric* to select a *nearest* state in the graph to connect to.
- Computing the control inputs and hence the trajectories that connect a newly sampled state to the nearest state.

Both steps translate to solving one or more optimal control problems, as the notion of *distance* in state space is the optimal cost-to-go between two states. Hence, the number of optimal control problems to solve exponentially increases with the size of the graph. Once a nearest state is determined, the corresponding optimal control inputs are used to generate the trajectory to connect to the randomly sampled state. Collectively, this leads to planning times that are prohibitively large, making sampling-based planning in state space practically challenging to use even in simulation experiments. Therefore, current research on motion planning in state space is focused on devising methods that can significantly reduce planning times in state space.

1.5.2. RELEVANT LITERATURE

The general problem of motion planning in state space was already an active topic of research before the inception of sampling-based methods. In the early 90s, Donald and Xavier [33] proposed an exact solution for time optimal motion planning in the state space of a point mass system governed by Newtonian Mechanics. This marked the first departure from planning in configuration space to planning in state space. Apart from being one of the first works in this domain, their work provided some key insights into the underlying complexity of the motion planning problem in state space when the dimensionality of the planning space is increased. The success of sampling-based methods in dealing with motion planning problems in higher dimensional configuration spaces inspired the works on sampling-based methods in state space [80, 51]. Since then, a majority of research efforts have focused on tackling the aforementioned challenges posed by the state space infrastructure.

The authors of [80] proposed a weighted Euclidean distance as a metric for the RRT approach in state space but were also critical about the choice, as Euclidean distance fails to adequately represent the (optimal) cost-to-go between state pairs. In their analysis and propositions of alternative metrics, they make a key remark that a reasonable approximation of the optimal cost-to-go could greatly improve the performance of RRT in state space. Since then, there has been a steady progress in the development of methods that present different ways to reliably approximate the distance metric in state space. As highlighted in [44, 57] these contributions can be roughly classified in two categories: (i) methods that propose different ways to account for system dynamics in metric computation, (ii) methods that limit the effect of a poor metric by using extrinsic qualitative information.

The first category of methods take inspiration from (linear) control theoretic principles to reduce the complexity of solving for the cost-to-go measure. For example, the authors in [68, 90] use the property of differential flatness in certain class of non-holonomic systems to transform the differential equations of motion to a set of polynomials in states and inputs. In [44, 105, 121, 143], the authors propose a distance heuristic in state space based on the

principles of linear optimal control where a linear or an affine quadratic regulator is used to compute the cost-to-go between state pairs. In [130], the authors propose a method to initially compute a local reachable set approximation of node(s) in the RRT and subsequently use Euclidean distance to choose the appropriate node to expand the tree.

The second category of methods focus on indirectly tackling the sensitivity of RRT growth to the distance metric. The main idea of these approaches is to collect qualitative information such as overlapping edges in the graph caused by poor selection of nodes and gradually reduce the chance of such nodes being selected [20]. In [19], collision information is used to prevent the selection of nodes that lead to expansions that are unusable. In [57], the authors propose a combination of both categories of methods by using system dynamics as well as environmental information.

It is pertinent to recall that fast and reliable approximation of the true distance metric in state space is the core motivation for the methods developed thus far. This is achieved via linearized or polynomial approximations of the actual (nonlinear) dynamics. However, as the authors of [44] point out, the benefits of such approximations drop off as the system complexities and nonlinearities increase. Thus, in the pursuit of speed, these methods compromise on reliability and as a consequence, further exploration of the potentials of planning in state space is hindered. Therefore, in the second part of this thesis, we focus on the development of a generic infrastructure where fast and accurate approximations of the true distance metric is possible without approximating the actual dynamics.

1.5.3. PROBLEM STATEMENT - PART II

In order to combine the benefits of sampling-based planning with optimal control so that motion planning in state space can be realized for practical applications, the following research questions are formulated:

3. How can we alleviate the computational demands of the distance metric computation for motion planning in state space? (Chapter 5)

4. How can we formulate the optimal control problem to quickly generate control inputs without compromising on the dynamical constraints? (Chapter 6)

The field of sampling-based motion planning in the state space is an active research topic and many current results (including those in this thesis) are on academic applications (simple pendulum swing-up). There is still quite some research to be done in this area in order to be able to make the step towards real-world industrial applications.

1.6. CONTRIBUTIONS AND THESIS STRUCTURE

The contributions in this thesis related to sampling-based motion planning in the configuration space are centered around the development of motion modules by making effective use of off-the-shelf open source software components. Below, the contributions from the first part of this thesis are briefly presented:

- Parameter tuning of path planning algorithms: With so many implementations available for path planning algorithms, it is a daunting task to make an appropriate choice of a planner and the corresponding implementation to be used for a specific problem.

We present a framework in this thesis to automatically tune the parameters of several path planning algorithms where a user can specify the desired performance metrics from a planning software framework. These results are presented in Chapter 2.

- Motion module design for industrial bin picking: Bin picking is a commonly encountered industrial application where different objects located in a certain environment are manipulated as a part of an order picking task. We present a framework for the design of a motion planning module using off-the-shelf software components so that the designed functionality is modular and reusable in similar applications. The module presented in this thesis was one of the components of the world championship winning robotic system at the Amazon Picking Challenge 2016[2]. The details of the motion module and the complete software framework of the entire robotic system constitute the contents of Chapter 3.

- Reactive collision avoidance for Cobots: Collaborative robots (or Cobots) are systems that can safely interact and closely work with human beings and are viewed as essential components in the Industries of the future. Motion modules that not only perform motion tasks (such as pick and place) but also ensure safety of humans in the vicinity are critical to the realization of such collaborative systems. To address this challenge, a framework was developed as a collective effort between various research organizations participating in the European Union Project *Factory in a Day*. The work involved in extending and integrating the previous contribution to this framework is presented in Chapter 4.

The second part of this thesis focuses on the challenges of motion planning in state space. Particularly, the critical steps of approximating a reliable (pseudo)metric and a fast steering function for the RRT in state space. We hypothesize that the principles of supervised learning could serve as tools to reliably approximate solutions to optimal control problems. The consequent gain in computation speed would thus enable the application of sampling-based motion planning in state space for practical applications involving manipulator motions. The contributions made in this direction are presented below:

- Distance metric approximation with supervised learning: A supervised learning-based approach to approximate (locally) optimal cost-to-go between state pairs is presented in Chapter 5. The core idea is to use an offline *training phase* to solve optimal control problems between several state pairs to a create a dataset that stores the state pairs and the corresponding costs. This dataset is subsequently used by a supervised learning algorithm to learn the mapping between state pairs and cost. The mapping is later used to predict costs during the actual run of the RRT algorithm. The results of the approach on the problem of pendulum swing up is presented including a speed up of up to 3 orders of magnitude relative to solving an optimal control problem at each incremental step of the RRT. While the results are on a simple system, the approach itself is generic and the results of using the approach on higher dimensional problems is also presented.

[2] http://www.robocup2016.org/en/events/amazon-picking-challenge/

- Steering input approximation with supervised learning: A fast approximation of the distance (pseudo)metric only solves the problem of determining the nearest neighbor in the RRT. Once a choice is made, there is a need to solve an optimal control problem to generate a trajectory between the nearest neighbor and the random state and this does contribute significantly to the overall planning time. An approach that builds on the previous contribution is presented in Chapter 6 to approximate the steering input to connect random state pairs which results in an order of magnitude speed up over state of the art approaches for motion planning in state space.

Finally, the thesis is concluded in Chapter 7 with a discussion on the contributions of this thesis and potential directions for future work.

I

SAMPLING-BASED PLANNING IN CONFIGURATION SPACE

2

AUTOMATED TUNING OF PATH PLANNING ALGORITHMS

A large number of novel path planning methods for a wide range of problems have been described in literature over the past few decades. These algorithms can often be configured using a set of parameters that greatly influence their performance. In a typical use case, these parameters are only very slightly tuned or even left untouched. Systematic approaches to tune parameters of path planning algorithms have been largely unexplored. At the same time, there is a rising interest in the planning and robotics communities regarding the real world application of these theoretically developed and simulation-tested planning algorithms. In this chapter, we propose the use of Sequential Model-based Algorithm Configuration (SMAC) tools to address these concerns. We show that it is possible to improve the performance of a planning algorithm for a specific problem without the need of in-depth knowledge of the algorithm itself. We compare five planners that see a lot of practical usage on three typical industrial pick-and-place tasks to demonstrate the effectiveness of the method.

Note: In this chapter, the word *configuration* has two meanings. The first being a configuration which is a vector used to specify a robot configuration. The second being a set of parameters used to specify an algorithm configuration. These differences are explicitly clarified in the relevant portions of the text.

The contents of this chapter have been slightly modified from the paper:
Ruben Burger, Mukunda Bharatheesha, Marc van Eert and Robert Babuška, *Automated Tuning of Path Planning Algorithms*, 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 4371-4376. The modifications are primarily to maintain consistency with the notations and symbols used in this thesis.

RECENTLY there has been a growing interest in the use of robotics software. With the increasing popularity of tools like ROS [115], MoveIt! [133], Open Motion Planning Library (OMPL) [135] and OpenRave [30], even hobbyists have now ventured into robotics. These tools have also played a major role in enabling end-users to leverage state-of-the art for real world applications.

2.1. TUNING OF PATH PLANNING ALGORITHMS

Development of novel planning methods with high theoretical merit makes up a large body of research within the motion planning community. Many of these methods are designed to solve a certain set of problems that have some specific characteristic. Consider Transition-based RRT (T-RRT) [56], which combines the exploration strength of RRTs with cost-map methods in order to guide the algorithms to paths that are low cost according to a specified cost metric. T-RRT manages to efficiently solve the posed problem, but how it translates to a different domain is unclear and difficult to estimate without a significant effort. A second example can be found in the set of informed planners like Informed-RRT and BIT* [40][39]. These planners work very well for scenarios in which the obstacles take a certain shape in the configuration space, but it is unclear how they can be leveraged for other classes of problems. They also have a number of configurable parameters that are difficult to understand without significant background knowledge. Another set of planners that use heuristics to guide the search are Kinodynamic Planning by Interior-Exterior Cell Exploration (KPIECE) methods [134]. These planners show good practical performance, but they use a discretization grid that makes it difficult to understand how the planning performance is affected by the configuration of the planning algorithm.

When presented with the problem of selecting a suitable planner for a given real-world problem, one faces a myriad of options. With several different available planning libraries and approaches like OMPL [135], SBPL [22] , CHOMP [118] and STOMP [61], it requires significant knowledge to determine which planning algorithms are suitable for each particular application. Choosing a good configuration for each planning algorithm is even more difficult. A configuration includes both *categorical* (for setting up the planner) and the *numerical* (for tuning) parameters of the planner. The number of parameters ranges for different planning algorithms from just one for RRTConnect [72] to twelve for RRT*, all of which may or may not have a significant impact on the performance. Especially the frequent use of heuristics make it very difficult to predict the behavior of an algorithm as the heuristics interact with each other in unpredictable ways. This makes manual configuration very difficult. In the case of OMPL, some of the parameters are calculated using the characteristics of the environment. While this is indeed beneficial, a larger improvement can be expected with a more rigorous tuning approach.

Typically, end users of motion planning software libraries adhere to ad-hoc heuristics to tune parameters of planning algorithms. The end result of such tuning would indeed result in a satisfactory solution, but there is no indication or feedback to the user if a better solution is possible. This limitation can be addressed with a structured and pragmatic approach to configure the planner. Ideally, it should be possible for an end-user to provide a geometric description of the manipulator and the scene, along with a set of typical problems to let an automated tuning algorithm provide the optimal configuration for each planner. Our work focuses on providing a basic framework to achieve this goal. We focus on the use of motion

planning software [135, 133] for robotic manipulators. Despite using a specific class of motion planning software to present our results, our approach itself is generic and is not limited by our choice. The following section provides a brief review of relevant research focusing on parameter tuning for planning algorithms.

2.2. RELATED RESEARCH

A general approach to algorithm tuning can be found in automatic algorithm configuration methods. These methods are optimization methods that are specifically designed to find the best performing algorithm configuration for a specific problem instance. They often do so by running the algorithm with a certain starting configuration and repeatedly selecting a better configuration based on the previous results. Historically, the problem of algorithm configuration can be abstracted to the Design of Experiments (DOE) approach [122], where a sequence of experiments are conducted to understand the relationship between (physical) experimental parameters and the associated outcomes. Typically, these relationships are established as regression models described by a probability distribution function mostly because, random errors, which are an integral part of physical experiments are well covered by probability distributions of different kinds. Further, related methods such as Design and Analysis of Computer Experiments (DACE) [124] are mentioned in [52] and the references therein. Here, the effort is more focused on computer-based (simulation) experiments which are mostly free of random physical errors. In this sense, methods such as in [124] are much closely related to our work in comparison to the DOE-based methods.

Algorithm configuration can be broadly classified in two main categories, local methods and model-based methods. Examples of methods that use a local search are ParamILS [55] and F-RACE [13]. For Sequential Model Based Optimization (SMBO) methods, a regression model is used to predict the performance of the algorithm in previously unseen regions. Gaussian process (GP) models are a common choice as a non-parametric regression model [52]. One disadvantage of GP models is that they do not deal with categorical input data such as boolean parameters. This is also a drawback with the aforementioned classical methods in [122, 124]. In [54] a method is introduced that uses a different model class for the regression named Sequential Model-based Algorithm Configuration (SMAC). SMAC uses random forests in order to handle categorical parameters. Random forests are collections of regression trees, which are known to offer good performance for categorical input data [54]. Furthermore, SMAC has been shown to deliver promising performance on a diverse set of expensive black-box problems [53].

These methods have been applied to a wide variety of problems, most notably to the tuning of solvers for computationally hard problems. These solvers are typically based on local or tree search, often have many tuning parameters that are very difficult to tune by hand. As an example; the mixed integer programming solver IBM CPLEX has 76 parameters relating to its search strategy [54].

More similar to our proposed application, automatic configuration methods have been applied to classical planning problems. An approach that is similar to our method can be found in [127], where SMAC was used to automatically tune the Fast Downward planning system. In [35], ParamILS was used to tune the topology, transition rules and parameter values of control software for a robot swarm. In [9], constrained Bayesian optimization is proposed as an optimization method that can be used to tune certain target parameters. Using

constraints on the optimization, it is ensured that the tested configuration does not lead to safety-critical system failures. In [67] a model-free optimization is used to find optimal PID parameters.

Configuration and tuning of configuration space path planning algorithms in particular seems to be a less highlighted area of research. In [95], an infrastructure that can be used for benchmarking different configurations was introduced, but the problem of how to select competing configurations for each planner is left unexplored.

Typically, many parameters of planning algorithms that we consider in this work are categorical. Therefore, we choose SMAC as algorithm configuration and parameter tuning method.

2.3. PROBLEM STATEMENT

Let $\mathscr{C} \in \mathbb{R}^n$ be the complete configuration space of a robotic manipulator. Denote by $\mathscr{C}_{\text{obs}} \subseteq \mathscr{C}$ the obstacle space, which consists of all configurations of the robot that are invalidated by collisions with either itself or the environment. Let $\mathscr{C}_{\text{free}} \subseteq \mathscr{C}$, the complement of $\mathscr{C}_{\text{obs}}$, be the set of all valid configurations in the planning space.

Define $Q_{\text{start}} \subseteq \mathscr{C}_{\text{free}}$ and $Q_{\text{goal}} \subseteq \mathscr{C}_{\text{free}}$ to be a set of configurations representative of the planning problem complexity. For example, in a pick-and-place application, Q_{start} would consist of pick configurations, and Q_{goal} would be place configurations. A planning query ψ is then defined by a configuration pair, consisting of a start configuration and a goal configuration:

$$\psi = \{q_{\text{start}} \in Q_{\text{start}}, q_{\text{goal}} \in Q_{\text{goal}}\}$$

The outcome of a planning query is a path which is a sequence of configurations.

The set of all possible queries derived from Q_{start} and Q_{goal} is denoted by Ψ. Let $c \in \mathbb{R}^+$ denote a user defined measure that indicates the performance of a planning algorithm on the entire set Ψ. For k start configurations and m goal configurations, Ψ will consist of $k \times m$ pairs of configurations. Denote by P, a target planning algorithm with i configurable parameters. The parameter settings for P is then denoted by a set of i-tuple:

$$\Phi = \{\phi_0, \ldots, \phi_{i-1}\}$$

where ϕ_i is used to denote the parameter value.

Finally, a planner parameter tuning request where planner P, with parameter settings Φ, is required to solve a planning query ψ such that the resulting path stays in $\mathscr{C}_{\text{free}}$ can then be written as:

$$P(\mathscr{C}, \Phi, \psi)$$

The problem is to find the parameter setting Φ that maximizes the performance c of planner P on the complete set of planning queries Ψ.

2.4. METHOD

To translate the planner configuration problem into a format that SMAC can tune, we specify three ingredients: The algorithm A, the performance measure c and the problem instance I.

SMAC starts by evaluating the default configuration of A on I, and returns a performance measure after evaluating c. Subsequently, the current configuration and corresponding

performance measure are used to update SMAC's internal model using:

```
SMAC.FitModel()
```

Next, SMAC will select a new configuration to be tested. This operation is denoted by:

```
SMAC.SelectConfigurations()
```


SMAC uses the random forests model to formulate an *Expected Improvement* function and uses a local optimization to find the configuration that maximizes this function.

After the most promising configuration has been selected, it is evaluated on the problem instance I. Whenever a result has a cost lower than the current best performing configuration the corresponding configuration will be stored as the *incumbent* configuration. Finally, after the runtime exceeds the *AllowedRuntime*, the incumbent configuration and the corresponding parameter values are returned.

The main challenge with using SMAC to configure a configuration space planner is how to formulate the problem instance I and set up the performance measure c so that the performance of the resulting configuration also improves the performance over the complete set of problems Ψ. This will be discussed in the following sections.

2.4.1. FORMULATING THE PROBLEM INSTANCE I

In order for SMAC to optimize towards a meaningful configuration, care should be taken in formulating the problem instance I as a function of the configuration space $\mathscr{C}$ and the complete problem set Ψ.

The full problem set Ψ was split into a tuning, Ψ_t, and validation set Ψ_v. Due to the random nature of the planners, it is not feasible to use the individual problems of the set as the problem instance. SMAC will not be able to fit an accurate model due to the high variance. By using the full tuning set as the problem instance I, the variance of the resulting cost is minimized. Furthermore, the whole set will be evaluated several times to further decrease the variance. In choosing the number of repetitions, a trade-off has to be made between decreasing the variance of the planning algorithm and allowing SMAC to quickly test new configurations.

2.4.2. FORMULATING THE PERFORMANCE MEASURE c

The performance measure c is another key element. The goal of the performance measure is to transform the output of a single problem query into a quantitative measure that can be minimized by SMAC. In many practical scenarios, the aim of tuning is to improve the solving percentage and decreasing the planning time. Focusing on these goals, a very effective performance measure was found to be the average computation time.

In order to limit SMAC spending too much time on poorly performing configurations, a planning timeout is configured for each planner. Whenever this timeout value is reached, a penalty value is used so that the regression model can still learn from the evaluation. The complete method is shown in Algorithm 1.

2.5. RESULTS

In order to validate our tuning method, it was tested on three different environments with two different robots: two environments for the 6-DoF Universal Robots UR5, and one for

Algorithm 1 SMAC Tuning

Input : Planner P, Tuning set Ψ_t with N queries,
Configuration space $\mathscr{C}$, Iterations n,
default configuration Φ_0.
Output : Incumbent configuration Φ_{inc}
$\Phi = \Phi_0$
$best = \infty$
while *SMAC.Runtime* < *AllowedRuntime* **do**
  $result = 0$
  for $i = 0, \ldots, N-1$ **do**
    $\psi = \Psi(i)$
    for $0, \ldots, n-1$ **do**
      Timer.start()
      $P(\mathscr{C}, \Phi, \psi)$
      $result$ += Timer.stop()
  if $result < best$ **then**
    $\Phi_{inc} = \Phi$
    $best = result$
  SMAC.FitModel($result, \Phi$)
  Φ = SMAC.SelectConfigurations()
return ***Incumbent***

the 7-DoF KUKA LBR iiwa 7 R800. The following planners were selected to be tuned:

- **KPIECE** This algorithm uses several heuristics in order to guide the search. These heuristics make it hard to predict its performance and make it an ideal candidate for automated configuration.

- **BKPIECE** For much of the same reasons, the bi-directional variant of KPIECE will also be considered.

- **BIT** While BIT* is actually an optimal planner, it can easily be configured to only calculate the first solution. As this eliminates all optimal properties of BIT*, this planner will be denoted as BIT (without the asterisk). With several configurable parameters, BIT is another interesting candidate for using the SMAC tools for tuning the parameters to a given problem requirement.

- **RRTConnect** As RRTConnect only has a single parameter, it is not expected to gain much from tuning. However, it is worthwhile including RRTConnect as it is a widely adopted planner in more practical applications, known for short computation times and high solving percentages.

- **BiTRRT** Although very similar to RRTConnect, it is interesting to see whether the extra parameters that can be tuned for BiTRRT have a greater impact on its performance.

The experiments were conducted on a PC with a Intel i5-3470 CPU at 3.20 GHz and 8GB of RAM running Ubuntu 14.04. To model the environment ROS Indigo was used in combination with Moveit!. The planners that have been tested are part of the OMPL planning library.

2.5.1. UR5 SIMPLE PICK-AND-PLACE PROBLEM

The first UR5 problem represents a pick-and-place scenario where the robot is to rearrange objects on a table. For these experiments, Ψ_t consisted of 20 problems that were each iterated 5 times for a total of 100 queries. The planning time was selected to be 1 s and SMAC was allowed 30 min to find the best configuration. The validation results were obtained by testing on a distinct validation set Ψ_v. Figure 2.1 shows a photo of the environment of the UR5.

With the start close to the surface on the lower corner of the table, and the goal states on the far corner, this can be considered a relatively easy problem with the movement of the UR5 mostly unobstructed. This allowed for SMAC to make several hundred calls to the planning algorithm, especially for faster planners.

Figure 2.1: Photo of the first target problem environment for the UR5 manipulator.

Table 2.1 shows the averaged total results for the complete validation set. The path length is represented as the average 2-norm in the configuration space of the manipulator and is included to serve as an indicator for the planning algorithms ability to find short paths.

As expected, RRTConnect and BiTRRT have not improved much. For KPIECE, BKPIECE and BIT, the results are more striking. With computation times significantly improved and finding shorter paths, it is safe to conclude that the tuned configuration shows better performance than the default.

For BIT, the percentage of solved problems was increased from 92 % to 99 %. Closer inspection of the results shows that BIT was struggling with one of the problems in particular, only managing to solve about 40 % within the allocated time. After tuning, BIT managed to solve this problem in over 95 % of the cases.

Table 2.1: SMAC results for the first UR5 problem

Planner	Runtime [ms]	Solved [%]	Path length [2-norm]
RRTConnect	36.3	100	7.3
RRTConnect - SMAC	36.0	100	7.2
BiTRRT	46.8	100	7.1
BiTRRT - SMAC	45.8	99	7.2
BKPIECE	254	99	7.8
BKPIECE - SMAC	108	99	7.4
KPIECE	95.7	100	7.8
KPIECE - SMAC	40.3	100	7.5
BIT	102	92	9.0
BIT - SMAC	52.2	99	8.9

2.5.2. UR5 DIFFICULT PICK-AND-PLACE PROBLEM

The second scenario for the UR5 is a more difficult one. Aside from being in a constrained environment, the UR5 is fitted with a 20 cm long vacuum tool. The problem scenario is depicted in Figure 2.2.

With significantly longer planning times, SMAC could do fewer iterations in the allocated time of 30 min. Instead of 5 iterations of 20 problems, this scenario was tuned on 4 iterations of 25 problems. The start and goal states were selected as realistic picking and placing states, with the tip of the vacuum tool well inside the stow bin that can be seen on the right.

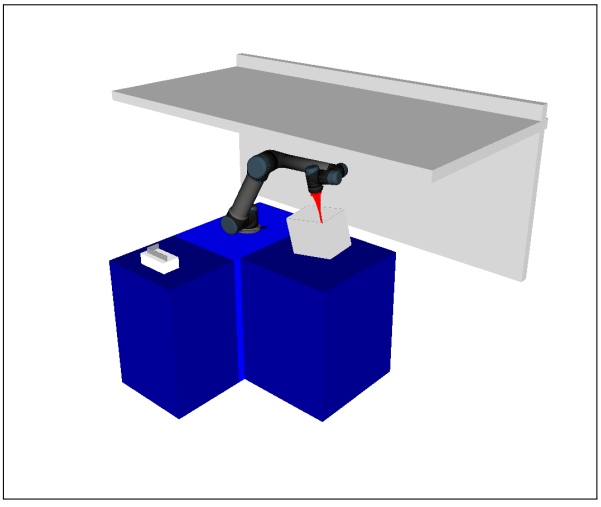


Figure 2.2: Model of the second, more complex problem environment for the UR5 manipulator.

Table 2.2: SMAC results for the second UR5 problem.

Planner	Runtime [ms]	Solved [%]	Path length [2-norm]
RRTConnect	165	95	10.9
RRTConnect - SMAC	146	96	10.2
BiTRRT	183	96	10.6
BiTRRT - SMAC	172	96	10.3
BKPIECE	751	45	12.6
BKPIECE - SMAC	450	94	10.6
KPIECE	443	79	11.8
KPIECE - SMAC	337	90	10.6
BIT	286	80	9.7
BIT - SMAC	266	81	9.2

Consider Table 2.2 for the results of tuning on the second scenario. As with the first scenario, the improvement of RRTConnect and BiTRRT is rather small. An impressive tuning result can be seen when considering BKPIECE, whose solving rate was increased from just 45 % to 95 %.

2.5.3. KUKA LBR IIWA 7 PROBLEM

A problem was designed for the KUKA LBR iiwa 7 manipulator that requires maneuvering to and from different poses inside a cabinet. The problem scenario is shown in Figure 2.3. With an extra degree of freedom and several start and goal states that are difficult to reach, this presents a more challenging problem than the previous two.

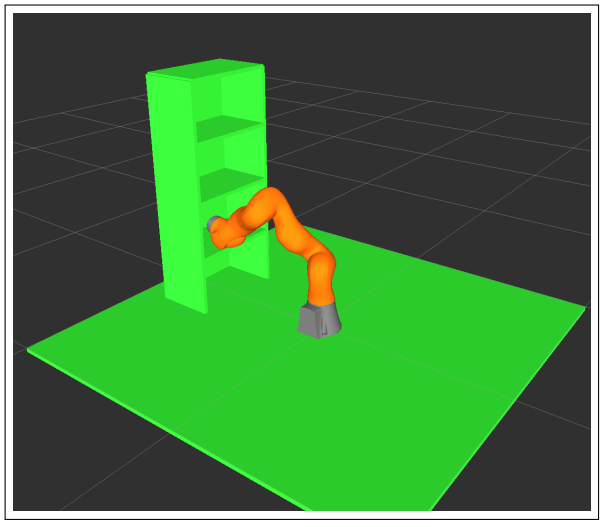


Figure 2.3: Model of the environment for the KUKA problem.

In order for SMAC to be able to test a configuration within reasonable time, the number of iterations were decreased and the planning timeout increased to 2 s. See Table 2.3 for the benchmark results after tuning.

Table 2.3: SMAC results for the KUKA problem

Planner	Runtime [ms]	Solved [%]	Path length [2-norm]
RRTConnect	725	100	6.5
RRTConnect - SMAC	687	100	6.6
BiTRRT	851	99	6.7
BiTRRT - SMAC	807	99	6.9
BKPIECE	1868	17	18.3
BKPIECE - SMAC	1930	18	16.2
KPIECE	1399	68	7.2
KPIECE - SMAC	613	99	6.0
BIT	1239	54	8.5
BIT - SMAC	871	84	6.8

Comparing the results of this scene with the UR5 scenes, it is obvious that all planners require more time to find a feasible plan but some (BIT and BKPIECE) seem to suffer more from the extra degree of freedom. The fact that BKPIECE even got *worse* (in terms of runtime) than before tuning, is related to the planning timeout and discussed in Section 2.6. As with the previous scenarios, a significant improvement can be observed for KPIECE and BIT.

2.6. DISCUSSION

A running theme in the previous section that is substantiated with the tabulated results is that automated tuning and configuration is beneficial for all the presented algorithms. In this section, we highlight some of the important aspects from the obtained results. A comparison of planner parameter values before and after SMAC tuning for three planners which showed significant improvement in performance on all three experimental setups is shown in Table 2.4. Consider for example, the KPIECE planner. We can clearly see that along with the `goal_bias` parameter (which is a common candidate for ad-hoc tuning), the use of SMAC tuning also ensures the rest of the parameters are tuned. As a consequence, these parameters collectively lead to the improved performance seen in the experimental results. However, an interesting analysis would be to also study the impact of each parameter individually and use the corresponding information to formulate better performance measures.

It is important to note that the performance improvement is not necessarily uniform over all the problems considered. Complex problems typically take longer to be solved and as there is no scaling between problems in the test set, configurations that improve on these more difficult problems are prioritized. With the BIT* algorithm, we have observed a number of problem queries that could not be solved even after tuning. While tuning improved the performance on the other problems of the set, these specific problems remained either very

Table 2.4: Comparison of planner parameters before and after tuning

	Before	After
RRTConnect		
`range`	default‡	3.33
BKPIECE		
`range`	default‡	0.3
`border_fraction`	0.9	0.47
`failed_expansion_score_factor`	0.5	0.24
`min_valid_path_fraction`	0.5	0.31
KPIECE		
`range`	default‡	4.12
`goal_bias`	0.05	0.2
`border_fraction`	0.9	0.96
`failed_expansion_score_factor`	0.5	0.71
`min_valid_path_fraction`	0.5	0.7

‡ *default* indicates the value computed by MoveIt! for a given problem instance.

difficult or outright unsolvable to BIT. A SMAC performance measure that puts a bigger penalty on unsolved queries can potentially be used to seek for configurations that do manage to solve all queries, but this is a topic for further research.

In the SMAC manual, it is advised to allow SMAC at least 300 to 400 attempts at finding a good configuration. In addition, the best results are achieved when SMAC is run several times with different starting configurations. However, in the 30 min of allowed planning time that was used in the experiments, SMAC often only had time for about 100 configuration tests. As SMAC is often used for algorithms with many more parameters than these planning algorithms, perhaps these requirements can be relaxed somewhat, but further research into the configuration of SMAC and the problem instance is encouraged.

It is pertinent to note the importance of correctly specifying the performance measure and planning timeout. For the KUKA scenario, BKPIECE only managed to solve 17 % of the queries when the timeout was set for 2 s.

Running SMAC with a 2 s timeout means that for each query that goes over the average even very slightly, a planning time of 2 s is reported. This means that SMAC gathers very little information on the actual performance of a configuration, and is not able to construct a good model. See Table 2.5 for different tuning results.

Lastly, the combination of MoveIt! and OMPL has been chosen specifically because of the seamless integration between the two software packages and the relative ease with which the combination allows one to use different planners for a given planning problem. However, we would like to reiterate that our approach itself is not limited to this specific software combination.

Table 2.5: BKPIECE tuning results with a variable planning time
Benchmark was run with 2 s timeout

Planner	Runtime [ms]	Solved [%]	Path length [2-norm]
untuned	1868	17	18.3
2 s tuning	1930	18	16.2
4 s tuning	1699	64	8.9

2.7. CONCLUSIONS AND FUTURE WORK

Our goal was to come up with a tool to enable easy tuning of planning algorithms for a given problem. To this end, we present the SMAC-based tuning method in our work and substantiate the achieved performance improvement in three realistic scenarios using industrial manipulators. Our primary goal with this work is to minimize the amount of background knowledge required to use planning algorithms, particularly in industrial robotic applications. Our long term goal is to further develop this approach in an extensive manner to include any dynamic parameters as well. Eventually, we would like to enable robotics end-users to transition from teaching task specific motions to robots to using planning algorithms to perform task specific motions.

3

LESSONS LEARNED FROM WINNING THE AMAZON ROBOTICS CHALLENGE 2016

What drew me towards team sport were the camaraderie and friendship. The chance to celebrate victory and success with a group of other people is something I have enjoyed doing.

Rahul S. Dravid

This chapter describes Team Delft's robot winning the Amazon Robotics Challenge 2016*. The competition involves automating pick and place operations in semi-structured environments, specifically the shelves in an Amazon warehouse. Team Delft's entry demonstrated that current robot technology can already address most of the challenges in product handling: object recognition, grasping, motion, or task planning; under broad yet bounded conditions. The system combines an industrial robot arm, 3D cameras and a custom gripper. The robot's software is based on the Robot Operating System to implement solutions based on deep learning and other state-of-the-art artificial intelligence techniques, and to integrate them with off-the-shelf components. From the experience developing the robotic system it is concluded that: 1) the specific task conditions should guide the selection of the solution for each capability required, 2) understanding the characteristics of the individual solutions and the assumptions they embed is critical to integrate a performing system from them, and 3) this characterization can be based on 'levels of robot automation'.*

The contents of this chapter have been slightly modified from the paper:
Carlos Hernández Corbato, Mukunda Bharatheesha, Jeff van Egmond, Jihong Ju and Martijn Wisse, *Integrating Different Levels of Automation: Lessons from Winning the Amazon Robotics Challenge 2016*, Accepted for publication in IEEE Transactions on Industrial Informatics - Special Issue on Recent Trends and Developments in Industry 4.0 Motivated Robotics Solutions, 2018.

THE Amazon Robotic Challenge (ARC) [5, 25], was launched by Amazon Robotics in 2015 to promote research into unstructured warehouse automation and specifically robotic manipulation for picking and stocking of products. Low volume, high-mix productions require flexibility to cope with an unstructured environment, and adaptability to quickly and cost-effectively reconfigure the system to different tasks. Current commercial solutions have mainly focused on automating the transport inside the warehouse, whereas only few solutions exist for the individual handling of the products [1], and are usually limited to one product type at a time [1]. Flexible robotics solutions are needed that benefit from current advances in artificial intelligence and integrate them with more dexterous and reliable mechanical designs for grippers and manipulators.

The integration of these robot technologies into an agile and robust solution, capable of performing on the factory floor, is itself a challenge. During a robot application's development design decisions need to be made, e.g. about feedback control vs. planning, that entail trade-offs between flexibility and performance. For example, in the first ARC edition in 2015, the winning robot used a feedback approach with visual servoing, achieving a robust pick execution that outperformed the competitors. However, the public media was disappointed about the general speed performance of the robots [96]. The average pick time for the winner was above one minute (~ 30 sorts per hour), while industry demands the ~ 400 sorts per hour achieved by humans [25].

There were two key ideas guiding Team Delft's approach to building the ARC 2016 robot: 1) reuse available solutions whenever possible, and 2) develop solutions to automate the system to the level required by the different challenges in the competition, making useful assumptions based on the structure present in the tasks.

To reuse available off-the-shelf solutions, Team Delft robot was based on an industrial manipulator and 3D cameras, and the robot software was based on the Robot Operating System (ROS) [115]. ROS provides tools and infrastructure to develop complex robotic systems, runtime communications middleware, and the ROS open-source community provides off-the-shelf components for many robot capabilities.

There is a variety of aspects that have been identified useful to characterize robotic systems [34]: modularity vs. integration, computation vs. embodiment, planning vs. feedback, or generality vs. assumptions. The dichotomy *planning vs. feedback* in robotics represents only two (important) classes in the spectrum of solutions. These range from open-loop solutions that exploit assumptions and knowledge about the task and the workspace at design time, to feedback strategies that use runtime information to drive robot's behavior and deal with uncertainties. After analysis and reflection on the ARC robot development experience, different *levels of robot automation* are proposed in this chapter to characterize the design solutions. In Team Delft's robot design, the different solutions were chosen to automate every part of the robotic system to the level required. Different automation solutions render different system properties in terms of flexibility, performance, and adaptability.

Section 3.1 discusses the requirements posed by the ARC 2016 competition scenario, and analyses the challenges it poses to robot perception, manipulation and task planning. In Section 3.2 the levels of robot automation are presented, and used to explain Team Delft's robot concept in Section 3.3. The performance of the system is discussed in view of the levels

[1] e.g. see work of Profactor GmbH. at https://www.profactor.at/en/solutions/flexible-robotic/handling/

of automation in Section 3.4, and some lessons learned are reported. Finally Section 3.5 provides concluding remarks.

3.1. MANIPULATION IN THE AMAZON ROBOTICS CHALLENGE

The Amazon Robotics Challenge (ARC) stems from a broader and fundamental research field of robotic manipulation in unstructured environments. The two tasks for the 2016 challenge [4] involved manipulating diverse, small sized products to pick and place them from an Amazon shelving unit (*the shelf*) structured in twelve bins, to a temporary container (*the tote*), as is illustrated in Figure 3.1. We begin this section by providing further technical details of the challenge followed by a comparative analysis of the challenge to relevant scientific problems.

Figure 3.1: The products in the Amazon Picking Challenge 2016 in the tote and in the bins of the shelf, from [4].

3.1.1. THE AMAZON ROBOTICS CHALLENGE 2016

The challenge for the year 2016 was titled Amazon Picking Challenge and consisted of two tasks to be autonomously performed by a robotic system: **The Picking Task** consisted of moving 12 products from a partially filled shelf, into the tote. Some target products could be partially occluded or in contact with other products, but no product would be fully occluded. Each of the 12 bins contains exactly one target product as well as any number of non-target products and every target product is only present in a single bin. The tote is initially empty in this task.

The Stowing Task was the inverse: stow the contents of the tote (12 products) into the bins of the partially filled shelf. The products in the tote could be partially or completely occluded below other products. There is no target location for the products in the tote, but different score for stowing them into more cluttered bins.

In both tasks the robot had 15 min to fulfill the order, which is specified by a task file, and report the final location of all the products in an analogous output file. The task file contained information of what products were located in which bin or tote and it identified the target products. The task file did not contain information about the physical location of products within their respective container. The target products could be handled in any order and all the product could be moved to any bin, as long as the final contents of each bin and the tote were correctly reported in the output file. The performance of the robot

is evaluated by giving points for correctly placed items and subtracting penalty points for dropping, damaging or misplacing items. Here misplacing an item is defined as incorrectly reporting its location in the output file. The amount of points for a specific operation would depend on the difficulty of the object and the cluttering of the bin. The time to accomplish the first successful operation would be the tiebreaker.

3.1.2. MANIPULATION IN UNSTRUCTURED ENVIRONMENTS

The ARC scenario is representative of the challenges in handling applications in a warehouse or the factory floor. The robot has to perform a few simple tasks in a closed environment, but it is only semi-structured. Unlike dynamic, open environments where autonomous robots have to cope with unbounded levels of uncertainty, here it is limited. However, uncertainty is still present, in the target products characteristics, their position and orientation, and the workspace conditions.

The set of 39 product types used in the competition includes books, cubic boxes, clothing, soft objects, and irregularly shaped objects. They were chosen to be representative of the products handled on a daily basis at an Amazon warehouse. They presented realistic challenges for perception, grasping and manipulation: reflective packaging, wide range of dimensions, and weight or deformable shapes.

The products are stored mixed in any position and orientation inside the shelf's bins, partially occluding each other, sometimes placed at the back. Bins could be too cluttered even for a human hand to easily pick the target item. The shelf construction with metal parts and cardboard divisions resulted in wide tolerances and asymmetries. Besides, the narrow opening of the bins (21 cm × 28 cm) compared to their depth (43 cm) limited the maneuverability inside, and caused difficult dark lighting conditions. The highly reflective metal floor of the bins contributed to the challenges for any vision system. In addition, the position of the entire shelf had ± 3 cm tolerance.

The variety of shapes, sizes and weights of the objects also posed an interesting challenge for object manipulation. This variety entailed studying and applying different grasp synthesis methods such as pinch grasping [104, 26] and suction grasping, successfully used in the previous edition of ARC [34]. The limited space for manipulation discarded cage grasping strategies. Regarding grasp synthesizing, despite the extended literature on grasping of unknown objects [114], the fact that the products were known well in advance made fine-tunned heuristics promise much better performance, as early tests demonstrated.

3.2. LEVELS OF AUTOMATION

The performance and flexibility of a robotic application depends on the assumptions and design decisions made to address uncertainty. A proper understanding of these decisions is specially important in robots that perform more traditional automation tasks, but with challenging flexibility in not so-well structured environments, such as the ARC. For this a characterization of the solutions in *levels of robot automation* is proposed, based on the experience gained developing Team Delft's robot. Our model is inspired by that of Parasuraman *et al.* [102]. While that framework supports decisions about which functions to automate, and to what extent, with a focus on the human interaction factor, the model presented here applies to the case of full automation of a function. It provides a basis

for deciding how to automate those functions, in view of the uncertainty present and the reliability required. The main criteria to differentiate automation solutions is the timing of the information used to drive the behavior of the system. Assumptions are prior information that is used at design time to determine a certain behavior, reducing the flexibility of the system, but generally optimizing its performance. On the other hand, closed control loops in a robotic system use runtime information to adapt the system behavior to the actual circumstances on-the-fly.

In traditional automation, the environment and the task are fixed and assumed perfectly modeled. This allows to fix at design time the sequence of operations and open-loop robot motions. Uncertainty is reduced to minor allowed deviations on product placement and geometry, which are accommodated for by robust and compliant hardware designs. This *'level 0'* automation allows to maximize the motion's speed leading to very high performance. However, it has no flexibility: the robot's behavior is fixed during design, no runtime information is used to adapt to deviations.

Open-loop automation solutions typically include error handling mechanisms, so that the robotic system can accommodate for foreseeable events during its design. These *'level 1'* solutions introduce sensing capabilities in the system to verify *a posteriori* the result of actions. For example in suction-based object handling the pressure in the suction cup can be checked to confirm a successful grasp or to detect dropping the object.

In *'level 2'* of robot automation, more advanced and rich perception is used to drive the robot behavior at runtime, following the so called *sense-plan-act* paradigm. The complete sequence of control actions is computed based on a predefined model of the world and initial run-time sensor information that accommodates any run-time uncertainty. A typical example is a vision-based solution that locates target objects. The limitations of this approach are well known in robotics and artificial intelligence fields [16].

In feedback control (*'level 3'*), action is dynamically computed at a certain frequency using runtime sensor information. Often, the target variables cannot been sensed at the desired frequency, or they are not directly accessible at all. In these cases, an estimation is used to close a feedback loop at runtime. The controller of a robot manipulator, closing a control loop for its joint state, is an example of *'level'* present in Team Delft robot.

Finally, a *'level 4'* solution uses predictions in addition to the current sensor information to optimize its response to an uncertain environment. This is the case in systems that use any flavor of model predictive control [17], in which a more or less complex model of the system dynamics is used to optimize the control action based on the predicted evolution.

The selection of the level of automation for each specific problem in a robot manipulation application implies a trade-off between flexibility, performance, and resources. In the following sections the Team Delft robot for the ARC 2016 is discussed, explaining the rationale for the different technological approaches chosen following the model of *'levels of automation'*.

3.3. ROBOTIC SYSTEM OVERVIEW

Based on the analysis of previous experiences in the ARC [34, 25], Team Delft's solution targeted three key performance objectives to maximize scoring in the competition: be able to complete all the tasks, robustness and speed. The design approach to address them was to develop the robot automation level more efficient considering the uncertainty challenges in

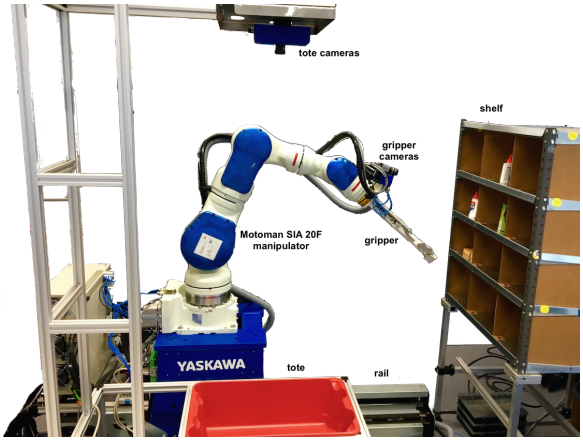


Figure 3.2: Team Delft robot setup in the APC workcell.

the tasks, and to reuse existing hardware and software components.

3.3.1. SYSTEM REQUIREMENTS

The performance objectives were decomposed into specific system requirements for robot manipulation. Completing the picking and stowing tasks requires the robot to handle all the products in any position in the shelf and the tote. This entails the following requirements:

Req. 1: to recognize and locate any of the products in any place inside the shelf or the tote.

Req. 2: to reach any location inside the bins with enough maneuverability.

Req. 3: to achieve and hold a firm grasp on all different products.

Robustness is a must in real-world applications that need to perform with almost no downtime. In the competition only one attempt[2] was allowed for each task, so any failure leading to the robot stopping or crashing is fatal. Speed is also critical for production systems. In Team Delft's strategy, speed allows the robot to perform several attempts to pick a target difficult to grasp, and also move other objects for clearance, during the 15 min allowed for each task. This simplifies the manipulation actions needed, leading to a more robust system.

3.3.2. ROBOT CONCEPT

Team Delft's robotic system is shown in Figure 3.2. It is based on an industrial manipulator mounting a 3D camera to scan the contents of the shelf's bins and a custom, hybrid gripper featuring a suction cup and a pinch mechanism. An additional fixed camera allows scanning the tote contents. The selection of this hardware will be justified together the explanation of the main robot functionality each device supports, in subsections 3.3.3, 3.3.4 and 3.3.5.

The ARC competition requires the robot to operate autonomously to complete tasks defined in a computer file that defines the current inventory of the shelf and the tote, and, for

[2] A reset was allowed: the task could be restarted from the beginning but with a penalty [4].

the pick task, the target product in each bin to be placed in the tote. Team Delft's solution for the picking and the stowing tasks is to decompose them into a plan of pick&place operations that is sequentially executed by the robot arm.

TASK PLANNING

For the Picking Task, the picking order of each target is computed to maximize scoring and minimize risk, considering i) the points for each product, and ii) system's confidence to handle each product, from experimental results, and iii) the need to move occluding objects (see rightmost flow in Figure 3.3). This way the plan of pick&place operation for all targets in the task is created. The plan is updated at the end of each operation according to its success or any fallback triggered (see failures in the right side of 3.3), as will be explained in section 3.3.6. For the stowing task, a simple heuristic selects as a target the detected product that is closer to the tote opening, since all the contents in the tote have to be stowed.

PICK&PLACE

The pick&place operations required to handle the products in the competition have a fixed structure in a closed, semi-structured environment: pick target *(*X) that is located in bin *Y* or the tote, and place it on the tote or bin *Y'*. Therefore a *'level 2'* robot control solution was designed, consisting of a sequence of actions that follows the sense-plan-act paradigm. The general steps and the main actions depicted in Figure 3.3 are as follows:

Sense

The system uses the 3D camera information of the target's container (bin or tote) to: i) detect and estimate the 6D pose of the item, and ii) obtain collision information of the container to later plan the motions, in the form of a filtered Point Cloud of the cluttering of the container. In the Pick Task scenario, the robot has to previously move to obtain the camera information for the target bin. Additionally, the actual position of the bin is also estimated, for a more detailed collision model of the environment. In the Stow task, a Point Cloud model of the tote is used, since its pose is perfectly known.

Plan

Using the estimated pose of the target item and its known characteristics, the system computes the grasping strategy and a grasp candidate (a pose for the gripper to pick the object). The sensed PCL collision information is integrated in an octomap with the known environment geometry, stored in the Universal Robot Description Format (URDF) to generate collision-free plan to approach the grasp candidate pose, pick the product and retreat from the shelf.

Act

The previous motion plan is executed as feedforward action, including gripper configuration and activation on the way to pick the item. Pick success is confirmed if possible with the pressure in the suction cup. If so, the robot moves to drop the item in the tote using offline generated trajectories.

Thanks to the structure of the environment, to place the products a robot automation *'level 0'* solution was designed that uses pre-defined motions to drop them either in the tote or the shelf's bins in a safe manner to comply with the competition rules[3]. In the case of placing the items in the tote, it is divided in 6 predefined drop locations, and the task planner

[3]The other award-winning entries exploited the rules by dragging products out of the shelf to fall into a tote strategically located below it. This resulted in targets 'successfully' handled, but fewer points due to penalties.

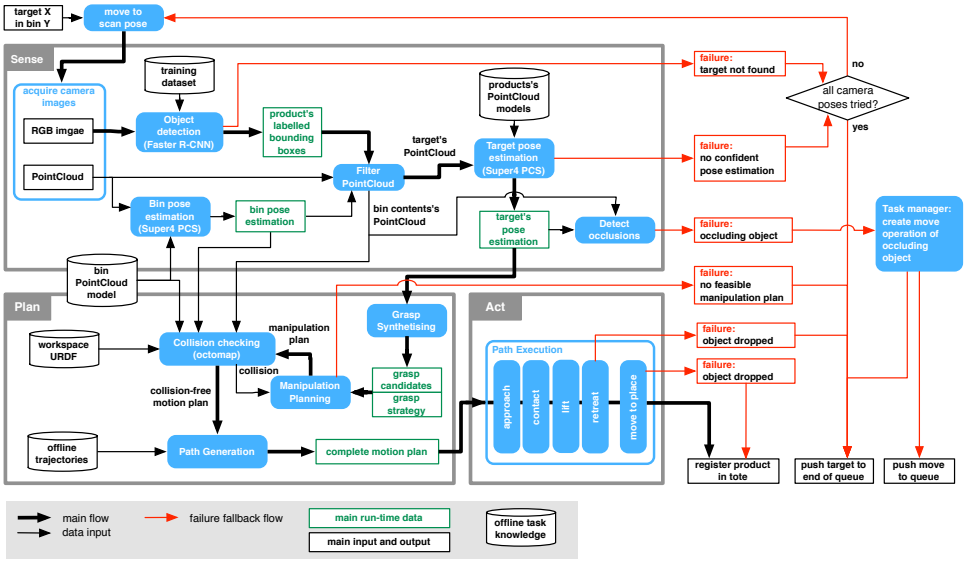


Figure 3.3: Schema of Team Delft's sense-plan-act workflow for picking a product *X* from the shelf's bin *Y*. The sense step consists of: a) detection of the target item and estimation of its 6d pose, and b) obtain collision information inside the bin, in the form of a PointCloud.

logic makes sure that: i) no heavy products are dropped where fragile items have been placed and ii) no more than 3 products are dropped in the same location, so that the objects do not protrude from the tote. In the case of moving occluding items to another bin, the task planner logic selects the lest cluttered bin from those that no longer need to be accessed (to keep the environment static). The robot moves to a fixed location in that bin, making use of the assumption that thanks to gravity any cluttering is in the lower part, and any standing items will be toppled inwards.

Sections 3.3.3 to 3.3.5 describe the solutions designed for all the robot capabilities required for the previous actions, grouped into object detection and pose estimation, grasping and robot motion, including the main hardware and software components involved.

3.3.3. VISION-BASED PERCEPTION

To address Req. 1, the robot needs to recognize and locate the objects captured by the camera, knowing what the object is and where it locates in the image.

CAMERAS

To scan the bin an industrial camera system is mounted on the gripper. It includes an Ensenso N35 3D camera that provides low noise point cloud data, and an IDS UI-5240CP-C-HQ high-definition camera that provides RGB images. An array of LEDs improves robustness to lighting conditions. A similar system is fixed on a pole to scan the tote.

OBJECT DETECTION

The object detection module takes RGB images as input and returns a list of the object proposals. Each proposal contains the label and the location, determined by a bounding box

enclosing the object. The proposals are ranked in descending order based on the confidences, varying from 0 to 1. The proposal with the highest confidence for the expected item was counted as the detected object.

One of the main difficulties for object detection is that each instance varies significantly regarding size, shape, appearance, poses, etc. The object detection module should be able to recognize and locate the objects regardless of how objects from one category differ visually. A model that has high capacity and can learn from large-scale data is required. Deep Neural Networks are renowned for it high capacity, especially the Convolution Neural Networks (CNN) have recently shown its ability to learn large-scale data efficiently, improving the benchmark performance of large scale visual recognition problems significantly since 2012 [123].

Ross et. al. [43] adopted the Convolutional Networks for classification and selective search [140] for region proposal in their framework, region-based Convolutional Networks (R-CNN), achieving a performance improvement by a large margin. One of the limitations of their work is that it took about 47 s[4] to create the region proposals and predict the object categories, for each image. The following studies [42, 120] accelerated the processing cycle time to 198 ms by applying the CNN more efficiently, including extracting convolutional features for the whole image and sharing CNN for region proposal and classification. The resulting method is referred to as Faster R-CNN [120].

The significant processing speed acceleration of the Faster R-CNN consolidates the basis of nearly real-time object detection in robotic applications. This is the reason Faster R-CNN was adopted in Team Delft's solution to detect objects in both the shelf and the tote of the ARC.

In the ARC setup, objects were placed in two different scenes, either in a dark shelf bin or a red tote. Therefore, two different models were trained to detect objects in the two different scenes. A total of three sets of RGB labeled images, were used for training:

Base Images of all the products were recorded automatically. Objects were put on a rotating plate against a monochrome background, and a camera was attached to a robot arm, taking images from different angles. Annotations were generated after automated object segmentation by thresholding. Images were augmented by replacing the monochrome background with random pictures after creating labels. This set contains in total 20K images.

Bin Images were taken for objects randomly placed in the bin. Annotations were created manually. This set includes 672 images.

Tote Images were taken for objects randomly placed in the tote. Annotations were created manually. This set contains 459 images.

The pre-trained weights of the VGG net [132] were used as initialization for the convolutional filters while the other filters were initialized with small random values drawn from a Gaussian distribution [45].

Given the three different sets of labeled images, five models were trained in a two-step strategy:

[4] All process timings run on one Nvidia K40 GPU overclocked to 875 MHz as provided in papers [42, 120].

Table 3.1: Evaluation of the Convolutional Neural Networks

Network	Bin test mAP	Tote test mAP
Base Model	16.1 %	7.9 %
Bin Model (bin data only)	82.9 %	-
Bin Model	85.7 %	-
Tote Model (tote data only)	-	90.0 %
Tote Model	-	92.5 %

Step 1 Trained the initialized model with all the images from the Base set, obtaining a *Base model*.

Step 2 Fine-tuning the Base model with scene-specific images, the Bin set and the Tote set, obtaining scene-specific models, a *Bin model* and a *Tote model*.

The first model is the Base model. For both the Bin and Tote models, two different models were trained. One model uses only the data from the respective environment (omitting step 1 of the two-step strategy), whereas the second model is obtained by refining the Base model with the environment specific training set; applying both steps.

The trained models were tested using 10 % of the Bin set, and Tote set, respectively, as two test sets. The test sets were excluded from the training procedure. An object proposal was counted as correct if it had more than 50 % of the area overlapped with the corresponding annotation. Average Precision (AP) were used to evaluate the ranked list of object proposals for each item category, and the mean over the 39 categories, known as Mean Average Precision (mAP), were used as the performance measure of the models. The Mean Average Precision varies from 0 to 1, and higher mAP indicates that the predictions match better with the annotations.

The result of this evaluation can be seen in Table 3.1. From this it is observed that the best results are obtained by refining the generic Base model with environment specific data. The Bin model was used in the ARC 2016 for the picking task and the Tote model was used for the stowing task.

OBJECT POSE ESTIMATION

While object detection localizes objects in 2D, handling the target objects requires knowing the 3D pose of the object with respect to the robot. The chosen approach separates pose estimation in two stages: global pose estimation and a local refinement step.

Global pose estimation was done using Super 4PCS [89]. Since this method compares a small subset of both a model point cloud and the measured point cloud for congruency, it can obtain a good global estimation of the object pose. This global estimation is then used as an initial guess in applying Iterative Closest Point (ICP) [10] for a close approximation.

For these methods, point clouds without color information were used. While it has been suggested [89] that using color information is possible in Super 4PCS, no analysis of its effect on the pose estimation performance was reported in that study. Furthermore it would have required obtaining accurate colored point cloud models of the objects, while for

most objects a simple primitive shape can be used to generate a point cloud model if color information is ignored. For some more elaborately shaped objects (a pair of dog bones and a utility brush for instance), a 3D scan without color information has been used as a model.

It should be noted that the Super 4PCS method inherently uses the 3D structure to obtain a pose estimation. Lack of such structure in the observed point cloud leads to suboptimal results. For example, observing only one face of a cuboid object could lead to the wrong face of the model being matched to the observation.

3.3.4. GRASPING

Team Delft's approach to grasping and manipulation is to simplify the problem to a minimum set of action primitives, relying on the following additional requirements:
Req. 4: a suitable grasp surface is always directly accessible from the bin opening that allows to grasp and retreat holding the product, and no complex manipulation inside the bin or the tote is needed. This way the *'level 2'* assumption of environment invariance holds.
Req. 5: the system should be able to find a collision-free path to grasp the product, and a retreat path holding the product.
Req. 6: the gripper is able to pick and robustly hold any of the 39 product types, compensating for small deviations, minor collisions of the held product, inertia and gravity effects on grasp stability.

GRIPPER

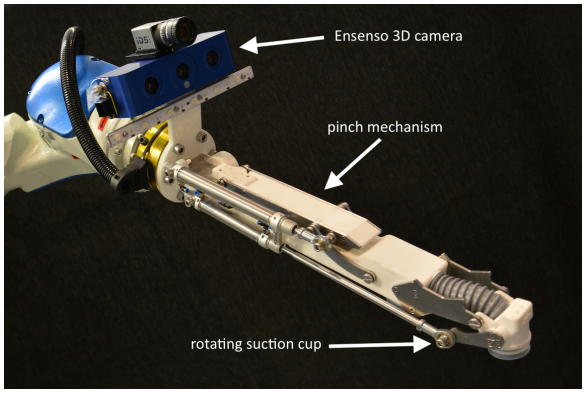


Figure 3.4: Team Delft gripper.

A custom hybrid gripper (Figure 3.4) was tailored to handle all items in the competition (Req. 6). It includes a suction cup based on low vacuum and high volume for robustness, and a pinch mechanism for the two products difficult to grasp with suction: a 3 pound dumbbell and a pencil holder made out of wire mesh. The gripper's 40 cm length allows to reach all the way inside the bins without the bulky robot wrist entering, and its lean design facilitates maneuverability inside the reduced space. The suction cup features a 90° rotation to provide an extra degree of freedom. This allows using the front surface of the object facing the robot for grasping, facilitating Req. 4. All the moving mechanisms on the gripper are controlled via pneumatic actuations using the digital I/O pins of the robot controller.

(a) Top suction (b) Front suction (c) Side suction (d) Pinch

Figure 3.5: The different grasp strategies possible with Team Delft's custom gripper.

This design strategy ensured two important aspects from a practical perspective. The gripper can be quickly dismantled and reassembled from the rest of the robot and the need to use delicate electronic control equipment that are vulnerable to transportation is eliminated.

GRASP STRATEGIES

The grasping strategy is chosen based on the type of product, its pose and the surrounding cluttering, from these primitives: *front suction*, *side or top suction*, and *pinch* (see Figure 3.5). The chosen primitive is parametrized to the target product by computing the grasp candidate pose and an associated manipulation plan, using *a priori* knowledge and runtime information.

GRASP SYNTHESIZING

The grasp candidate is a pose that if reached by its end-effector allows the robot to grasp the product activating the gripper according to the chosen primitive (by generating suction or the pinch mechanism). For non-deformable products the grasp candidate is generated using heuristics for the different product types based on geometric primitives and the structure of the workspace, as detailed in [50]. Since a 3D pose estimation is not possible for deformable products, grasp candidates are obtained using the surface normals of the detected object's point cloud, and ranked according to their distance to its centroid.

3.3.5. ROBOT MOTION

The motion motion module is responsible for moving the end-effector to all the poses needed along the sense-plan-act behavior, fulfilling Req. 2 for reachability, Req. 5 for grasping and the requirement for speed.

CHOICE OF ROBOT

To chose a robot manipulator that could execute all required motions, a workspace reachability analysis using MoveIt! [133] was conducted. The robotic system designed consists of a 7 degrees of freedom SIA20F Motoman industrial manipulator mounted on a rail perpendicular to the front side of the shelf. The resulting 8 degrees of freedom allows to reach all the bins with enough maneuverability.

The motion problem was simplified using two assumptions about the workspace uncertainty:

1. outside the shelf, the environment is static and known, and the task involves a finite set of poses to scan the bins and the tote, and to access them, so motions can be pre-planned offline (*'level 0'* solution).

2. inside the shelf and the tote the environment is also static but unknown. However, it has some structure due to: the walls, the given condition of products not laying on top of each other, gravity, and in the case of the shelf the assumption of one surface accessible from the bin opening.

MOTIONS OUTSIDE THE SHELF AND TOTE

Using assumption 1, a *'level 0'* solution was implemented to implement the motions needed outside the shelf and tote. Around 30 end-effector poses were pre-defined, and collision-free trajectories between all of them were planned offline.

MANIPULATION PLANNING

Using the second assumption, the manipulation strategy was designed from a motion perspective as a combination of linear segments to *approach*, *contact*, *lift* and *retreat*. These segments are computed online from the grasp candidate poses using cartesian planning. Collisions are accounted for using the shelf's or tote's 3D model and online information of the surroundings by generating an occupancy octomap from the scanned point cloud.

PATH GENERATION AND EXECUTION

Finally, the offline trajectories and the manipulation segments are stitched into a complete time parameterized motion plan. This process optimizes the timely execution of the motions. It allows for custom velocity scaling to adapt the motions to safely handle the heavier products. This process also synchronizes along the trajectory the timely configuration (to the desired strategy), and later activation of the gripper to pick the target product, and the verification of the pressure sensor in the gripper after retreat. Finally, the resulting trajectory is executed by the robot manipulator, also controlling the gripper[5].

3.3.6. FAILURE MANAGEMENT

Special focus was given to the overall reliability of the robotic system. The system can detect a set of failures during the sense, plan and act phases and trigger fallbacks to prevent a stall. For example, if the target product cannot be located, or estimate its pose, different camera poses are tried. If the problem persists it will postpone that target and move to the next one. A failed suction grasp is detected by checking the vacuum sealing after execution of the complete grasp and retreat action. In that case, it is assumed that the item dropped inside the bin and retries the pick later. If the vacuum seal is broken during the placement in the tote, the item is reported to be in the tote, since it can actually be the case, and there is no gain for reporting dropped items. For a pinch grasp, the system could only validate the complete pick and place operation by checking the presence of the picked item in the image from the tote.

3.4. DISCUSSION

The Amazon Robotics Challenge is a valuable benchmark for robot manipulation solutions. It provides interesting indicators to measure the advantages and limitations of the different robotic solutions. Following we discuss the performance of Team Delft robot using the automation levels framework presented in section 3.2 and analyzing the different trade-offs of using run-time feedback or design assumptions to address the uncertain conditions.

[5]For additional design and implementation details of the motion planning module, please refer to Appendix A.

3.4.1. EVALUATION

Table 3.2 summarizes the results of Team Delft's robot in the competition to win both challenge tasks [50]. The detailed analysis of performance in Table 3.3 shows that the design achieved the system requirements targeted in speed and reliability while addressing the uncertainty conditions presented in section 3.1.2. However, the system presented some limitations that affected its performance specially in the picking task.

Table 3.2: Result summary of the ARC 2016 Finals.

Team	Total score	successful targets	misplaced items
Picking Task			
Team Delft	105	9	3
PFN	105	9	1
NimbRo Picking	97	10	4
MIT	67	6	2
Stowing Task			
Team Delft	214	11	1
NimbRo Picking	186	11	2
MIT	164	9	0
PFN	161	10	3

DETECT AND LOCATE ALL THE PRODUCTS

The solution for object detection based on deep learning proved highly reliable and fast (avg. 150 ms). It is a *'level 1'* solution that takes additional images from fixed viewpoints on the fly if needed. The solution is robust to varying light conditions, including the dark locations at the back of the bins and the reflections at the front due to products packaging and the metal floor, at the cost of requiring large amounts of training data. On the other hand, it is highly reconfigurable: training the model for a new product requires only a few dozens of images and a few hours. This makes this approach very attractive for tasks where the arrangement of the products is unknown, but sample information can be easily generated.

The pose estimation of the target based on Super 4PCS is less reliable, and its speed had higher variance, due to which a time limit to 4 s was added to trigger fallback mechanisms. Speed could be improved with a GPU implementation of the Super 4PCS algorithm. The main problem for the open-loop solution implemented was the scarce point cloud information obtained for certain situations, strongly affected by lighting conditions and packaging reflections. The *'level 1'* fallback mechanism to take additional images from fixed locations was not an improvement. A possible improved *'level 2'* design would use the information from an initial pose estimation to plan a new camera pose. However, in many cases the pose estimation error considered the target in impossible or unreasonable orientations. A more promising enhancement is then to use more application heuristics during design, for example assuming that gravity limits the possible orientations of an object.

STABLE GRASP FOR ALL PRODUCTS AND ORIENTATIONS

Team Delft's grasping solution was able to pick all 39 items in the 2016 competition, in most orientations and bin locations. The *'level 1'* solution to grasping and product handling, based on a custom gripper, achieved a robust and fast performance for most of the products. The high-flow, low-vacuum combined with a compliant suction cup proved robust to different product surfaces and misalignments (90 % success rate). Additionally, it embedded grasp success sensing, providing an interesting standalone *'level 1'* solution. The main problems of the suction mechanism were: inadvertently grasping two objects, and the stability of the grasp for large, heavy products. The first problem could be improved by verifying the grasped products with the tote camera. The second problem is partially addressed by manipulation planning with custom heuristics to orient the product after grasping.

The pinch mechanism is less reliable (50 % success rate for the dumbbell). Its lack of compliance demanded a higher accuracy than that provided by the pose estimation module. Additionally, it is a *'level 0'* standalone solution with no feedback on the success of the operation.

REACH AND MANEUVER ALL LOCATIONS

Robot motion is critical in manipulation applications in relation to speed and collisions. Regarding speed, the overall *'level 2'* solution designed allowed to optimize the robot motions during design, achieving a high performance only limited by the grasp stability and safety[6]. As an indicator, Team Delft robot achieved an average cycle time of 35 s, compared to more than a minute for the feedback-based winning solution in 2015 [34].

In relation to reliability, Team Delft solution combined offline information in the 3D models of the workspace and the runtime point cloud information from the 3D cameras to avoid collisions. Avoiding collisions guarantees not modifying the structure of the environment, thus facilitating the application of an open-loop *'level 2'* solution. However, it is more limited for handling cluttered situations, where it can be unavoidable to touch objects next to the target. Additionally, it is more sensitive to perception errors and scarcity of data, as is the case of the shelf, where a limited range of viewpoints is possible.

OVERALL SOLUTION

The overall *'level 2'* sense-plan-act solution achieves a high-performance when the core assumption of an accessible grasp surface holds. This is the case in the stowing task, a standard bin picking application, thanks to gravity and task conditions (workspace geometry and product characteristics).

Bin cluttering presented the main difficulty to the *'level 2'* approach, making the collision-free requirement hard to address. Avoiding it by moving items around posed the disadvantage that an unknown number of pick and place actions could be needed to remove all the objects occluding the grasp of the target item. On the other hand, the simplification to a *'level 2'* system using only two primitive actions allowed us to optimize the required robot motions for speed, by exploiting the structure of task and environment as described in section 3.3.5.

[6] Due to the competition setup the robot speed limits were set to a safe 0.5 factor of its nominal maximum joint speed.

Table 3.3: System performance summary

44 pick & place operations attempted in the Picking and Stowing finals. 38 operations on target items, and 6 to move occluding items.			
Successful		25	11.35 s (avg) sense&plan 22.41 s (avg) act (robot motions) 33.76 s (avg) total pick&place execution time
Failed	recovered	3	target not detected or pose not estimated
		10	no motion plan found
		6	target dropped after grasp
	penalties	1	target dropped outside the bin
		1	non-target shoved outside the bin

System fatal errors	Stall condition due to no more feasible targets. Emergency stop due to gripper crushing object.

Another disadvantage of our *'level 2'* approach was the limited compliance with runtime uncertainty and its need for accuracy. The localization of the target products has to be precise to 5 mm, as well as the collision information. The hardware selection regarding the cameras and the gripper design proved that it is critical to simplify the control solution.

In relation to workspace uncertainty, the solution proved robust to deviations in the shelf's geometry, they being due to construction or to the 3 cm displacements allowed by the competition rules. These uncertainties were compensated for by the *bin pose estimation* procedure performed during the sense cycle. This approach turned out to be a significant factor in reducing the overall cycle time for the *first* pick which also was the *tiebreaker* to decide the winner if teams tied on points.

FAILURE MANAGEMENT

The possible failures of the system are: i) the product cannot be picked ii) product is dropped, iii) critical collision leading to equipment or product damage. These failures could arise from any of the core modules of the Team Delft-APC system namely Vision, Grasping or Motion. While exhaustive testing of all failure modes on the final system was difficult to perform due to practical reasons, some of the failures observed while testing and the actual run in the final competition are listed in Table 3.3.

The nature of the failures caused by the core module Vision were fairly consistent in the products and the situation that caused them. However, the failures caused by the core modules Grasping and Motion were difficult to reproduce as they were caused by IK solutions that would put the last few joints of the robot in a singular configuration while performing specific grasps (such as Bubble mailer leaning on the side walls of the top left or right corner bins). This was mainly caused by the numerical optimization based TRAC-IK solver used for the grasp motion planning. This non-determinism could have perhaps been reduced by using a fixed seed for the TRAC-IK solver, but, we did not have the time to implement and test this solution before the challenge.

The error handling mechanisms described in 3.3.6 provided for reliability when the product cannot be picked, by either retrying it under different conditions (new camera images), or postponing that target. When the product is dropped, appropriate action or reporting was coded using favorable assumptions about the result of the action. *'Level 1'* mechanisms for error handling specific to the application are unavoidable. However, general methods and tools that allow to capture them in a scalable and maintainable way are critical, such as the state-machine tool SMACH used by Team Delft.

3.4.2. LESSONS LEARNED

Overall Team Delft's approach to design the level of automation required for each problem proved to be successful, outperforming concurring and previous editions' entries in the Amazon Robotics Challenge. The main lessons learned relate to the value of open-loop solutions, how deep learning can contribute to them, and the need for collisions in manipulation.

In semi-structured, closed environments, the proper combination of open-loop solutions and hardware tailoring based on adequate assumptions provides the best performance. An exemplary case that proves that exploring simplifying hypothesis with open-loop solutions is worthy are the deformable products in the Amazon competition. Initially, they seemed to pose problems to locate, given the lack of a 3D model for them, and even more difficulties to manipulation planning. However, detection worked flawlessly with enough training data, and the compliant and powerful suction gripper made precise localization and careful manipulation unnecessary.

To address the limited flexibility and corner cases, Team Delft integrated two different solutions that exploit application knowledge at design time to tackle runtime uncertainty. For grasping, manipulation, and error handling, heuristics were programmed. They provide a simple and powerful solution easy to implement and inspect. However, their practical implementation presents important problems. First, they require intensive work by experts. They are hardly reusable. Finally, they do not scale: in complex applications such as the Amazon Robotics Challenge, where there are many robot technologies and components connected, modifying or adding new task-level heuristics in the design becomes unmanageable. The challenge remains to find practical ways to systematically encode knowledge, something the AI and cognitive robotics communities have been targeting for decades.

Deep learning techniques are a very promising solution to automate the generation of application-specific solutions embedding task-knowledge. Despite being an open-loop solution, automated training allows to easily accommodate for variations (bounded uncertainties), as well as to easily adapt it to new products or environments. In Team Delft's design, deep learning proved a very reliable and performing solution for object detection. Another successful entry in 2016 competition[7], as well as other results [82] suggest that it can be applied to grasping. However, the generation of training data in this case is very resource consuming. An intermediate, more feasible solution could be applying it to pose estimation.

Grasping from the bin poses more difficulties for Team Delft's open-loop solution. with many grasps being rejected due to collisions despite some of them being harmless or actually useful to grasp an object. Many results suggest that grasping and manipulation require to allow for contact, using techniques that incorporate force/torque information [34]. Feedback

[7] https://www.preferred-networks.jp/en/news/amazon-picking-challenge_result

solutions seem unavoidable for successful manipulation in cluttered environments. However, for improved performance it is desirable to limit their scope in the robot control by combining them with the better performing planning solutions. Current robot motion planning based on joint configuration space presents problems with grasp stability and does not allow for more flexible online planning based on force/torque feedback. Kinodynamic motion planning can overcome these limitations, but more research is needed for it to become a practical and feasible solution.

3.5. CONCLUSION

This chapter provides a practical discussion on the challenges for industrial robot manipulation for product handling, based on the experience of the authors developing the winning solution in the Amazon Robotics Challenge 2016. From this experience, we conclude: 1) the specific task conditions should guide the selection of the robotic solutions for an application, 2) understanding the characteristics of the solutions chosen and their relation to the task's conditions, embedded in multiple design decisions and assumptions, is critical for the performance of the overall system integrated from them, and 3) this characterization can be done according to 'robot automation levels', based on the use of information to address the task uncertainties during the development and runtime of the robotic system.

4

Dynamic Collision Avoidance for Collaborative Robot Applications

Current collaborative robot arms enable the creation of more flexible work cells. That is, they create the possibility to safely collaborate with human operators and thus augmenting productivity in tasks difficult for traditional automation. However, current solutions for safe interactions imply stopping the robot motion when a collision is detected. This reduces the productivity in an operational setup in which unintended, safe collisions can happen often. Active contact evasion by the robot arm is desirable so that the production process continues despite regular interferences and path obstructions. As a part of the European Union project Factory in a day (FiaD), multiple technologies have been developed such as, a proximity-sensing robot skin, a motion control framework based on proximity-sensing and a reactive path-planning solution. These technologies have been integrated into a dynamic-obstacle avoidance framework and successfully tested in simulation and laboratory set-ups. This chapter presents the obstacle avoidance solution that is realized with this framework for a collaborative pick and place application prototype. This prototype has also been successfully presented at two public events: RoboBusiness Europe 2017 *and at the European Innovation Summit* 2017 *and at the final demonstration of the FiaD project.*

The contents of this chapter have been derived from the paper:
Mukunda Bharatheesha, Nirmal Giftsun, Carlos Hernández Corbato, Gautier Dumonteil and Martijn Wisse, *Dynamic Collision Avoidance for Collaborative Robot Applications*, IC^3-Industry of the future: Collaborative, Connected, Cognitive, Workshop at the International Conference on Robotics and Automation (ICRA), 2017. The main difference to the paper is the inclusion of the completed results as opposed to the preliminary results mentioned in the paper.

THE desire for robotic solutions, particularly in the Small and Medium scale Enterprises (SMEs) is becoming increasingly prominent. Automation and robotics promise to deliver reduction on production costs and increase in productivity. However, traditional automation implies an investment prohibitive for SMEs, whose activities mainly involve small batches of production and high variety of products, for example, due to a seasonal nature of their operations. Concretely, tasks such as assembly, machine filling or packaging, can be automated with a robot in the work cell. However economic feasibility requires a reduction in robotization costs. The FiaD project [145] aims to achieve this by reducing system integration costs and installation time. The key idea is that the robot solution is flexible so that it can be quickly re-installed and configured to another temporary product line.

To achieve this flexibility and maintain acceptable levels of productivity, in the FiaD approach we propose to automate the easy 80 % of the tasks and leave the hard 20 % for human co-workers. Robot manipulators provide power, repeatability and extended workspace while the human operators provide flexibility and problem solving capacity. In addition, fenceless collaborative robots save space and installation cost. However, this approach requires a very high level of safety and agility; the robots should be aware of any obstacle, including dynamic obstacles such as its human co-workers, and be able to move to avoid contact. Whereas current co-bots guarantee safe contacts, they degrade the performance of the work cell because of stopping the production. In this chapter, we present one of the breakthrough innovations of the FiaD project: robot arms that are aware of all (dynamic) obstacles in their environment, and that respond by moving around these obstacles while still continuing their work.

The chapter is organized as follows. Section 4.1 describes the solution for dynamic obstacle avoidance that makes use of a proximity-sensing robot skin. In Section 4.2 the robot motion control architecture to incorporate the collision information as safety constraints to dynamically adapt the trajectory is presented. Section 4.3 presents the experimental results obtained in two different robot setups, one in simulation and one on a hardware prototype. Finally, in Section 4.4 we present our concluding remarks.

4.1. DYNAMIC OBSTACLE AVOIDANCE SOLUTION FOR COLLABORATIVE MANIPULATION

Current collaborative robot solutions guarantee safety, but they use obstacle detection to stop moving. Our dynamic obstacle avoidance solution is that of using obstacle detection to respond by moving around the obstacles while continuing to accomplish the desired tasks. The obstacles are detected by a proximity-sensing robot skin. Additionally, an integrated dynamic motion planning approach creates motion plans that fulfill various task specific constraints for typical industrial applications. For example the work cell 3D model is used to create a consistent model of the work environment, so that collision free trajectories are flexibly generated for different operations. The automatic consideration of these constraints drastically simplifies and speeds up the deployment of the robot.

The solution presented in this chapter relies on the innovative proximity-sensing Artificial Robot Skin (ARS) developed by the Institute of Cognitive Systems Systems in the Technical University of Munich [92]. This modular skin consist of identical ‘cells’ physically connected

forming skin 'patches'. These patches can be applied to cover the robot's links and joints, while being electronically connected to work as a single, modular robot skin. Each cell in the skin produces 4 modalities of perception: 3D acceleration, force, temperature, and distance. The multi-modal signals from the Artificial Robot Skin can be used to control the dynamic behavior of an industrial robot , for example to achieve compliant motions in a non- compliant robot manipulator [27]. These multi-modal signals can also be exploited to generate semantic representations [60] for teaching new task to the robot [31]. The ARS also features auto-calibration that allows to determine the kinematic chain of each cell to the robot base frame [91]. In this work, we use the distance provided by the optical proximity sensor in the skin, which is used to detect the obstacles around the robot.

An artist's illustration of our dynamic obstacle avoidance solution is shown in Figure 4.1. The robot motion control component generates appropriate motion commands for the robot controller to follow the trajectories required for a given task. The proximity-sensing skin that covers the links and joints of the manipulator, produces information regarding potential collisions. This information is used by the robot motion control module to adapt the robot motions on the fly to fulfill both constraints: following the current trajectory (with a certain tolerance) and avoid collisions. If the collision is unavoidable with local deformations of the current trajectory, the robot motion control module requests a (global) re-planning, which is performed on the fly by the reactive path-planner. The motion control then takes the end effector to the final goal pose using the alternative trajectory. The main functional modules of the system are discussed in the following sections of the chapter.

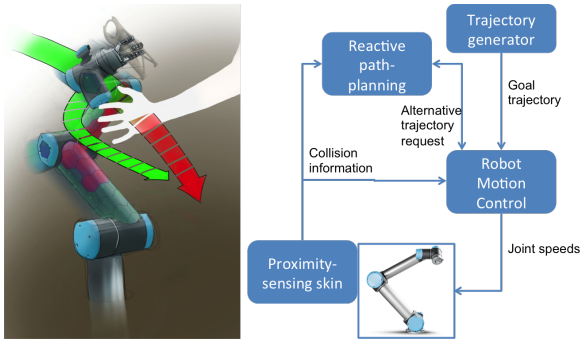


Figure 4.1: An artist's schematization of the FiaD Dynamic obstacle avoidance concept is illustrated on the left side. On the right, an overview of the main components of the solution.

4.2. ROBOT MOTION CONTROL USING PROXIMITY SENSING

The motion control is achieved using the Stack of Tasks (SoT) controller framework [88]. SoT employs a hierarchical jacobian control strategy eliminating the analytical inverse kinematics computation, thus making it a generic controller for all robot platforms. The controller's hierarchical nature allows the robot to handle multiple kinematic tasks simultaneously exploiting the kinematic redundancy of the robot. The controller's real time capability comes from the high computational speed of the state of the art Hierarchical Quadratic Programming (HQP) solver backing it.

A *task* basically is a control law that achieves a specific objective which can be a free-space task or just an inequality constraint that narrows down the workspace of the robot. The task function formalism is very well discussed in [125]. In the context of our work, tasks generally include robot joint posture task, collision avoidance task, joint limits task and so on. The SoT framework handles the task priorities hierarchically in real time to ensure there are no conflicts among tasks which is used to achieve dynamic obstacle avoidance without compromising on the main goal.

For example, let us consider a pick and place application in a collaborative environment. The primary goal for this application is to enable a robot to move to a (set of) desired pick and place locations repetitively. The pick and place locations can be defined as posture tasks in SoT. However, a higher priority task considering the collaborative nature of the environment is to avoid collisions with obstacles that could be humans, for instance. Typically such a task is modeled as an "Inequality task" and an eventual feasible solution (if one exists) is computed by the solver by exploiting the kinematic redundancy of the robot. In the jargon of motion planning and control, this behavior is similar to a *local planner*. However, it is likely that a feasible solution is not found due to the solver converging to a local minima[1]. In such a scenario, SoT can also be used to leverage the services of a global planner (see Section 4.2.1) from the current robot state to the goal so that an entirely new path is obtained which is free from collisions and consequently allowing all the specified tasks to be achieved in the order of their priorities. In Section 4.3, we present the experimental results of using the SoT controller on a practical setup and in simulation. The SoT controller has also been configured to work with the ROS-control interface. In all these setups, the proximity information from the artificial robot skin is used as an input to the collision avoidance task. In the following part, we briefly present the global path planner software framework that is used when the SoT controller hits a local minima.

4.2.1. REACTIVE PATH-PLANNING

The reactive path planning software framework is based on the industry grade KineoWorks™[2] [74] path planning library from Siemens in order to provide fast and reliable robot paths. This framework has also been seamlessly integrated into the ROS-ecosystem via a ROS package called `kws_ros_interface` which provides the planner implementations of KineoWorks as shared objects that are readily usable in ROS-based software via the `kws_ros_planner` ROS node.

Robot kinematic models are provided to KineoWorks in the Unified Robot Description Format (URDF) which is a ROS standard. Furthermore, KineoWorks also accepts the standard ROS representation of a `PointCloud`[3] for creating collision models of dynamic obstacles in the environment. In our work, point clouds are generated in two ways. In one scenario the point clouds are generated by a standard Kinect 3D camera that is observing the immediate environment of the robot. In the other scenario, the point clouds are generated from the proximity data obtained from the Artificial Skin. Finally, the collision detection for dynamic obstacle avoidance is performed using the Kineo™Collision Detector (KCD)[4].

[1] This is caused by the use of task jacobians and non-availiability of redundant degrees of freedom in the robot. For further details, please see [88].

[2] See http://www.plm.automation.siemens.com/en_us/products/open/kineo/kineoworks/index.shtml

[3] See http://wiki.ros.org/pcl

[4] See http://www.plm.automation.siemens.com/en_us/products/open/kineo/collision-detector/index.shtml

KCD performs 3D collision detection and minimal distance analysis between triangular mesh surfaces in assembly environments. KCD has been designed specifically to minimize memory usage and take advantage of parallel processing. The complete software architecture used in this work for the Dynamic Collision Avoidance functionality is shown in Figure 4.2.

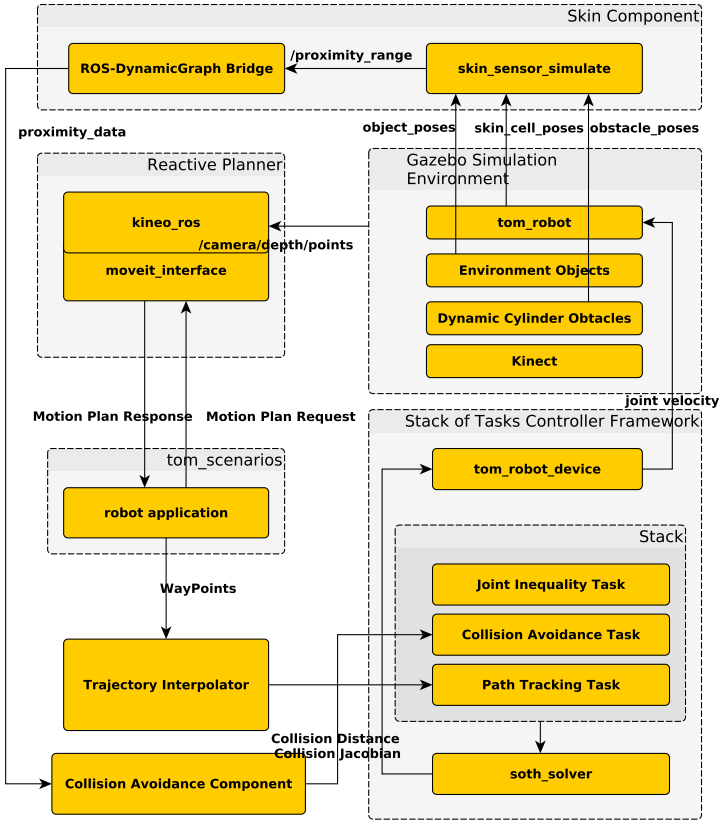


Figure 4.2: Dynamic collision avoidance software architecture for the simulated robotic setup in Figure 4.4.

In the following sections we present simulation and hardware results that have been obtained using the different functionalities described.

4.3. RESULTS

The results of evaluating the different functionalities are presented in two main categories. The first category includes results from individual evaluation of the different functional components on practical applications. The second category involves simulation and practical results of integrated evaluation of the functional components.

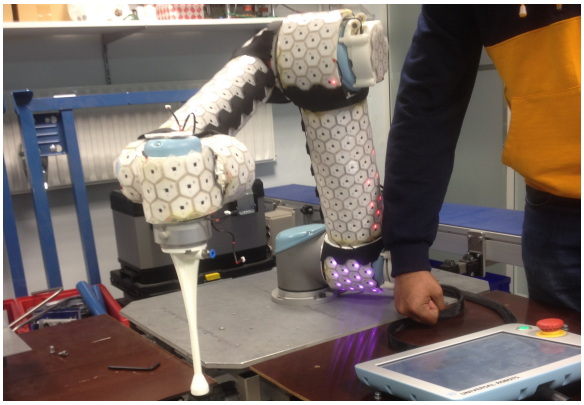


Figure 4.3: Robot setup showing Artificial Skin Cells being activated (with red LEDs) by obstacles (<= 6 cm).

4.3.1. INDIVIDUAL COMPONENTS

The *Artificial Robot Skin* (ARS) has been successfully deployed on a Universal Robots UR5 robot (see Figure 4.3). The ARS has been configured to provide proximity information related to obstacles in the immediate surroundings of the robot.The Stack of Tasks (SoT) controller has also been deployed and tested for achieving different postures on the setup in Figure 4.3.

4.3.2. INTEGRATED EVALUATION

The integration of all the components described earlier has been evaluated on a simulation of the orange sorting by Dean *et al* [28, 29] as shown in Figure 4.4.

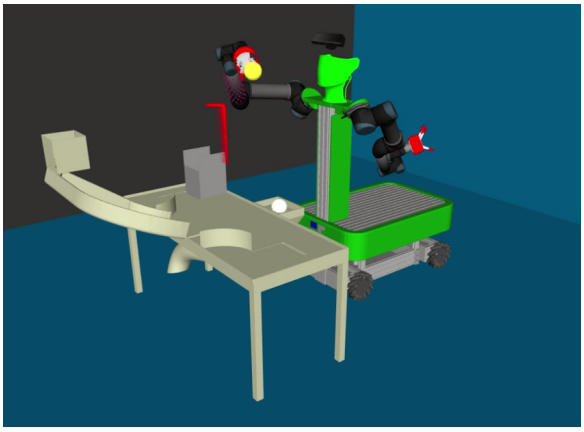


Figure 4.4: Orange sorting scenario in simulation.The red point cloud is a simulated obstacle.

The evaluation is done in a ROS based gazebo environment with the skin sensors simulated using the flexible collision library to project the distance between objects to sensor range measurements.These measurements are mapped to signals compatible in dynamic graph framework using a bridge component to allow its use in the SoT controller.

The collision avoidance component computes the point distance and jacobian of each and every skin cell configured essential to feed as an inequality constraint to the solver which backs the SoT controller. The planning component having the capability to plan with point cloud data has a MoveIt! [133] python interface to query motion plan requests. The response is a set of way points which is then linearly interpolated to instantaneous joint position commands to a path tracking task in the SoT. The SoT controller also has a python interface which makes it easy to design application scenarios. The combined use of a reactive motion planner and a hierarchical reactive SoT controller with skin data makes it a good candidate for dynamic obstacle avoidance applications in factory environments[5].

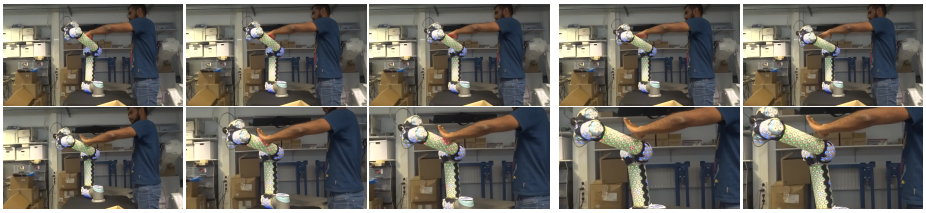


Figure 4.5: Collision avoidance task - The robot modifies its posture when an obstacle comes closer than 6 cm. L to R top row: As an obstacle (human hand) approaches closer to the skin cells on the forearm, the SoT controller computes commands such that the elbow joint moves in such a way that the desired distance is maintained. L to R bottom row: In this case, the base joint of the robot moves such that the desired distance to an obstacle is maintained.

The components have also been tested on a hardware prototype (Figure 4.5). As described in Section 4.2, the collision avoidance constraint is specified as an inequality task. In this context, the task stack is composed of two tasks in the order of their priorities: an inequality task for ensuring the generated control commands are within the specified joint limits and an inequality task for maintaining a distance of 6 cm or higher from any obstacle. In the SoT framework, this is specified by using a selection of skin cells located on the forearm of the UR5 robot, and each cell constitutes an inequality task with a priority just below the joint limits task. The number of such cells is required to be selected in advance. This is because, the SoT framework is currently limited to work with (task) jacobians of a fixed dimensions. Therefore, online modification in the size of the task stack for the HQP solver leads to numerical instabilities.

Once the basic collision avoidance behavior was realized, the trajectory following task was added to the task stack and it was assigned a priority lower than the collision avoidance task and naturally, the joint limits task. One of the practical issues we faced was that the proximity data beyond a distance of 6 cm was noisy and inconsistent for practical use. This impacted the possibility of using the KineoWorks™planner in the hardware setup. This was because, most of the valid proximity data (~6 cm) from the skin sensor was filtered out as self collisions by the reactive path planner and actual information about the robot surroundings could not be reliably obtained. Without the availability of any other sensor modality to acquire information about the surroundings of the robot, an alternative path was impossible to plan when the SoT controller was stuck in a local minima (see Figure 4.6).

[5] A video of this result is available in: https://youtu.be/uLStjR7mpOI

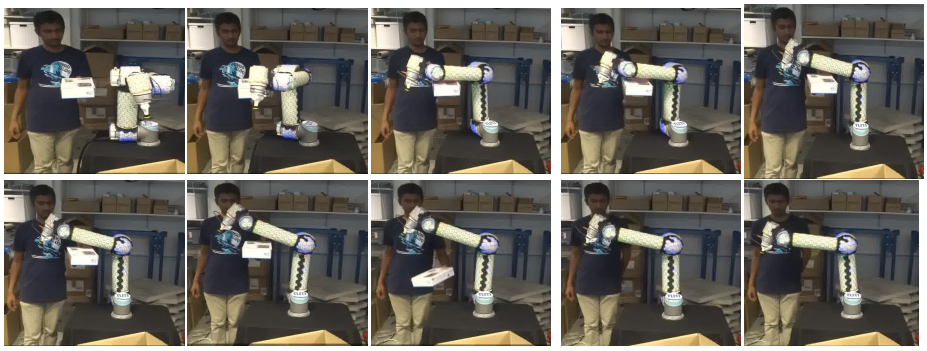


Figure 4.6: Collision avoidance task - Illustration of the local minima situation. L to R top row: An obstacle is introduced in the workspace close to the final joint configuration on the robot trajectory. However, the trajectory is not completely executed as the higher priority collision avoidance constraint cannot be satisfied at the desired final configuration. Without the availability of other sensing modalities to find alternative paths the robot is stuck in a local minima close to the final configuration. L to R bottom row: As the obstacle is removed, the collision avoidance constraint is satisfied and the robot completes the trajectory execution.

While the local minima situation is not desirable in an industrial scenario, the goal was to show a proof of concept of a possible solution. Therefore, these behaviors were integrated as a prototype application in a bin picking scenario. The reactive collision avoidance behavior achieved with this setup while performing a pick and place task is shown in Figure 4.7.

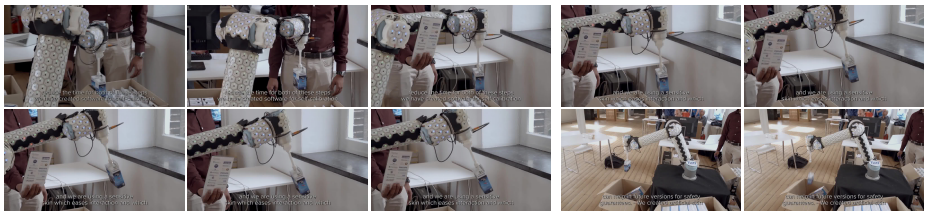


Figure 4.7: Collision avoidance task - Illustration of a successful alternate path generation behavior. L to R top row: After the robot has picked up an object, an obstacle is introduced along the robot trajectory to the object drop off location. L to R bottom row: The SoT controller computes an alternative path avoiding the obstacle while respecting the collision avoidance constraint and the trajectory execution task is successfully completed thereafter.

This prototype has been presented at multiple public events such as RoboBusiness Europe 2017 (RBE17), the final demonstration of the FiaD project and at the European Innovation Summit (EIS) at the European Parliament (see Figure 4.8).

4.4. CONCLUDING REMARKS

This chapter has presented the technologies that have been developed in the FiaD project to augment collaborative robot manipulators with dynamic obstacle avoidance. All these technologies: a proximity-sensing robot skin, a reactive path planning solution and a robot motion control strategy, have been validated in laboratory prototypes. Also, a preliminary prototype of an integrated solution based on these technologies has been tested in simulation and on real hardware. These results provide promising directions for making the next step towards deployment of collaborative robotic solutions for SMEs.

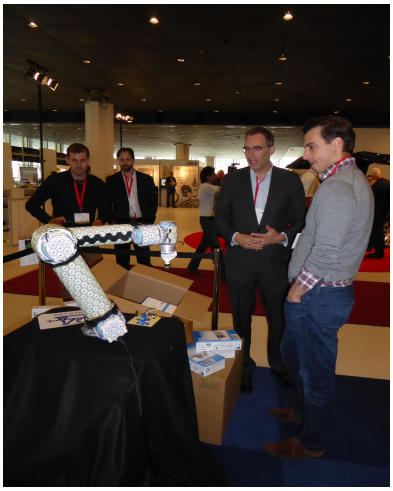


Figure 4.8: Hardware prototype for demonstrating the reactive collision avoidance behavior.

The integration and installation of advanced functionalities such as the dynamic obstacle avoidance solution presented poses three main challenges from the software point of view. The first is the integration of different components such as the skin driver, path planner and robot motion control. We address this challenge by adhering to the software development paradigm of the ROS-Industrial initiative. All the components discussed in this chapter have been successfully integrated with ROS.

A second challenge is the quality assurance and robustness of the integrated robot software. This is crucial in production environments, and is specially important in collaborative applications, where safety needs to be guaranteed. For this purpose an Automated testing Framework (ATF) has been developed [144] as a part of the FiaD project, which allows for the systematic testing of robot software components, which includes unit testing, simulation-in-the-loop testing and eventually hardware-in-the-loop testing. The tests can be automated and integrated in a centralized continuous integration system. Preliminary test have already been conducted with the components of the robot software system of this work, and the integrated prototype applications will be tested with ATF.

Finally, the third challenge is the deployment of the software. One of the main barriers to transfer solutions based on robot frameworks such as ROS to industry, and specially SMEs, is how cumbersome it is to deploy. As a part of the FiaD project, a Robot deployment toolbox has been developed [87], based on ROS, which can also be integrated with ATF. Due to time constraints, the testing of the deployment tools has not been performed on the robotic application presented in this chapter.

NOTES

- The hardware and software deployment of the Artificial Robot Skin on the robot setup used was done by Katharina Bulla and Florian Bergner with the support of Dr. Emmanuel Dean and Prof. Dr. Gordon Cheng from the Institute of Cognitive Systems, Technical University of Munich.

- The realization of the reactive controller on the robot hardware was achieved with strong support from Nirmal Giftsun and Florent Lamiraux from LAAS-CNRS, Toulouse, France.

II

SAMPLING-BASED PLANNING IN STATE SPACE

5

Distance Metric Approximation for State Space RRTs using Supervised Learning

Science is the art of the appropriate approximation. While the flat earth model is usually spoken of with derision it is still widely used. Flat maps, either in atlases or road maps, use the flat earth model as an approximation to the more complicated shape.

Byron K. Jennings

The dynamic feasibility of solutions to motion planning problems using Rapidly Exploring Random Trees depends strongly on the choice of the distance metric used while planning. The ideal distance metric is the optimal cost-to-go between two states in state space. However, it is computationally intensive to find the optimal cost while planning. In this chapter, we propose a novel approach to overcome this barrier by using a supervised learning algorithm that learns a nonlinear function which is an estimate of the optimal cost, via offline training. We use the Iterative Linear Quadratic Regulator approach for computing a (locally) optimal cost and learn this cost using Locally Weighted Projection Regression. We show that the learned function approximates the original cost with a reasonable tolerance and more importantly, gives a tremendous speed-up of a factor of 1000 *over the actual computation time. The effectiveness of our approach is shown on a pendulum swing up planning problem.*

The contents of this chapter have been derived from the paper:
Mukunda Bharatheesha, Wouter Caarls, Wouter Wolfslag and Martijn Wisse, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014, pg. 252-257. The main difference to the paper is the inclusion of additional results at the end of the chapter to further substantiate the validity of the proposed method. These results, although available at the time of writing the paper, were left out due to space limitations.

SAMPLING-BASED approaches for kinodynamic planning [80] have been an active subject of research in the past two decades. A commonality that lays beneath the myriad of approaches that have been proposed is the *tree* graph structure. Depending on how the nodes of a tree are chosen for further expansion, two specific research directions have emerged; choosing a node to expand based on *distance* to a randomly sampled node from the state-space [80] and choosing a node based on the *density* of nodes in a specific region of the state-space [51]. The former addresses the key question of formulating an appropriate distance metric while the latter focuses on appropriate methods to quickly estimate coverage densities. Typically, these approaches are well suited for offline kinodynamic planning. Due to the inherent exploration feature that is embedded in these approaches, these approaches solve the planning problem at a global level and thus guarantee goal reachability and avoid local minima.

5.1. KINODYNAMIC PLANNING

KINODYNAMIC motion planning has been a subject of extensive research in the past two decades and addresses the problem of finding a motion plan for a robotic system that adheres to the kinematic and dynamic constraints. Often, the state space of the system is the chosen candidate for kinodynamic planning as it naturally allows for accounting for the dynamical constraints of the system. However, a well known difficulty is that this problem is PSPACE-hard [78]. Hence, several practical solutions rely on the idea of sampling-based planning. Sampling-based approaches rely on constructing a tree graph structure with the states of the system as the tree nodes and the trajectories of the system between two states as the tree edges. Two approaches that have been extensively researched in this area are the Rapidly Exploring Random Trees (RRT) [80] and Probabilistic Road Maps (PRM) [51]. One of the main difference between the RRT and PRM approaches is the way in which an existing node in the tree is chosen for further expansion. In an RRT, the node to expand from is chosen based on the notion of a *distance* to a randomly sampled node in the state space. On the contrary, PRM-based approaches typically choose a node to expand based on a probabilistic weighting of the density of nodes in different regions of the state space. In our work, we focus on the RRT-based approaches. The RRT approach is briefly outlined in Algorithm 2. In the RRT algorithm mentioned above, N indicates the number of tree

Algorithm 2 RRT ((V, E), N)

for $j = 1, \ldots N$ **do**
  $\mathbf{x}_{\text{rand}} \leftarrow$ `Sample`
  $\mathbf{x}_{\text{nearest}} \leftarrow$ `Nearest` $(V, \mathbf{x}_{\text{rand}})$
  $(cost_{\text{xnew}}, \mathbf{x}_{\text{new}}) \leftarrow$ `Steer` $(\mathbf{x}_{\text{rand}}, \mathbf{x}_{\text{new}})$
  $X \leftarrow X \cup \{x_{\text{new}}\}$
  $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}}, cost_{\text{xnew}})\}$
return $G = (V, E)$

nodes to be generated, V is the vertex set and E is the edge set of the tree graph. The `Sample` procedure samples a random node from the state space. The `Nearest` procedure determines the nearest neighbor in the tree to the randomly sampled node based on an

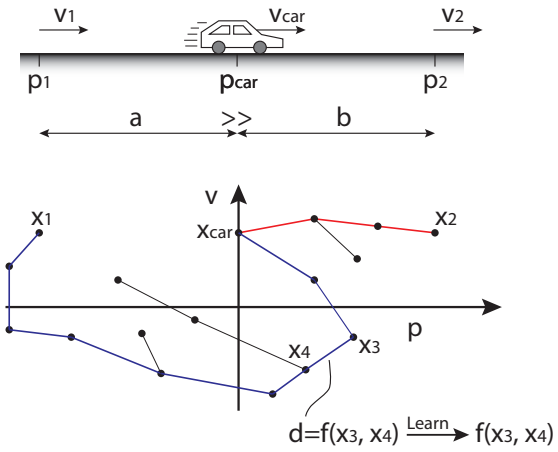


Figure 5.1: Simple depiction of the problem of distance metric in state space. While driving a car forward, it is generally easier to reach a point ahead of the car rather than reaching a point behind the car. The tree structures indicate a candidate solution that could be found by RRT.

appropriate distance metric. The `Steer` procedure attempts to connect the nearest neighbor to the randomly sampled node and also gives the corresponding cost to connect the two nodes. Typically, this is accomplished by forward simulation of system dynamics for a specified duration. If this attempt is successful, the randomly sampled node is added as the new node to the set of tree nodes. Otherwise, the state that is eventually reached by the `Steer` procedure is added as the new node to the set of tree nodes.

We explain the importance of distance metric in the state space with a simple example depicted in Figure 5.1. The position and velocity of the car are its states. Let us consider the state at time instant t_0 as (p_{car}, v_{car}). Let t_1 denote the time at which the state is (p_1, v_1), which is 10 m ahead of the car. For simplicity, we assume, $v_1 = v_{car}$. Similarly, let (p_2, v_2) indicate the state of the car at t_2, which is 10 m behind the car. In the context of state space, we consider (p_1, v_1) is *nearer* to (p_{car}, v_{car}) as the car is already moving forward and hence less work is needed. On the other hand, to reach (p_2, v_2) with $v_{car} = v_2$, we first reduce the car speed during which we move forward, come to a halt, drive backwards beyond p_2, again halt and then arrive at p_2 with velocity v_2. Hence, (p_2, v_2) is considered to be *farther* from (p_{car}, v_{car}) when compared to (p_1, v_1). In this sense, the *distance* in state space has the notion of a *cost-to-go* between two states. Furthermore, the optimal cost-to-go between two states is considered as the ideal distance metric in state space [80]. In fact, solving this problem exactly would eventually solve the overall motion planning problem [80].

In the simple car example described earlier, we can exactly solve for the optimal cost-to-go between states (denoted by a and b in Figure 5.1) in very short time and thus have a perfect distance metric. However, in systems with pronounced (and possibly non-linear) dynamical effects, computing the optimal cost takes a very large amount of time. This factor limits the possibility of using RRTs for online planning problems. It is also argued in [80] that approximations of these costs can significantly improve the feasibility of plans generated by RRT-based approaches. It is also well documented that the feasibility of the solution generated by state space RRTs is highly sensitive to the distance metric used [77, 44].

5.1.1. RELEVANT BACKGROUND

In literature, the choice of the distance metric has evolved significantly ever since the basic RRT implementation in [80] used the Euclidean distance. In fact, Euclidean distance is used in an indirect sense in [130] to assess the proximity of the randomly sampled node to the reachable set of a node in the tree and subsequently make the appropriate choice. In a subsequent work, Glassman and Tedrake [44] propose a metric based on the theory of Linear Quadratic Regulators (LQR) and affine dynamics of a non-linear system around a linearization point. They refer to their approach as the Affine Quadratic Regulator (AQR) design. While the work on AQR primarily focuses on the distance metric alone, the authors of [105] use the LQR theory not only to estimate the metric but also to use the resulting optimal policy for propagating the tree further. They also use the asymptotically optimal version of the RRT algorithm, namely RRT*, and hence establish the asymptotic optimality of the resulting plans from their approach.

A primary reason for the choice of LQR-based approaches for approximating the cost-to-go metric is the ease and speed with which they can be numerically computed. These approaches have shown to work with reasonable efficiency in [44, 105, 121]. The strategy of linearizing around sampled random points with either zero inputs or zero velocities or both is however, not a good approximation of the cost-to-go. In principle, such an approach would completely diminish the possibility of utilizing natural system dynamics. This problem is discussed in greater detail in Section 5.2.1. The authors in [44] discuss the situation of linearizing about points with non-zero inputs and velocities too, but mention that the performance of their approach drops off as nonlinear dynamics become prominent.

A solution to address this problem has been proposed in Li and Todorov [83], where nonlinear dynamics are linearized around a nominal trajectory instead of a point and an optimal cost-to-go is obtained iteratively by solving a modified LQR problem (iLQR). One of the primary reasons that this approach has not been popular in the sampling-based planning domain is the time needed to converge to the optimal cost. We defer further details on this approach to Section 5.2.1.

We recall from the car example in Section 5.1, that the nature of the distance metric in state space is nonlinear. In a mathematical sense, the distance metric is a nonlinear mapping $\rho : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^+$. The iLQR approach discussed earlier can also be considered one such mapping. Existing literature in the area of supervised learning indicates the possibility of approximating nonlinear functions. Our main contribution is showing that the cost calculated by iLQR can be accurately and efficiently approximated using supervised learning techniques. We use Locally Weighted Projection Regression (LWPR) [142], which allows for approximating nonlinear functions in high-dimensional spaces by using locally linear models. After off-line training on a dataset generated with iLQR, the on-line planning stage only requires a single evaluation of this model to approximate the distance between two points. In this way, we enable the RRT-based planning to avail the benefits of the iLQR approach without the increase in computation time. We substantiate the choice of LWPR learning over other supervised learning methods such as neural networks or SVM regression in Section 5.2.2.

The rest of the chapter is structured as follows. In Section 5.2, we formally present our problem along with a brief description of iLQR and LWPR algorithms. We follow it up with our proposed solution to combine the two approaches in Section 5.3. In Section 5.4, we

present the experimental results and highlight the resulting benefits. We briefly discuss our results in Section 5.5 and finally conclude in Section 5.6.

5.2. PROBLEM DESCRIPTION

Let us consider a robotic system with the following nonlinear dynamics:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{5.1}$$

where $\mathbf{x}$ is the state vector of the system and $\mathbf{u}$ is the input vector. Given the dynamics described by Eq. 5.1, the Rapidly Exploring Random Tree (RRT) algorithm allows for finding a motion plan to steer the system from an initial state $\mathbf{x}_i$ to a final state $\mathbf{x}_f$. A main ingredient for generating a feasible plan via the RRT approach is the distance metric $\rho: \mathscr{X} \times \mathscr{X} \rightarrow \mathbb{R}^+$. The distance metric is called at every iteration of an RRT. As described earlier, computing the metric in state space requires a significant amount of CPU time.

Our work proposes a solution that significantly reduces the distance metric computation time. We use a combination of the optimal control approach Iterative Linear Quadratic Regulator (iLQR) and a supervised learning approach called Locally Weighted Projection Regression (LWPR) to achieve the reduction in computation time. We briefly describe the two approaches in the following.

5.2.1. ITERATIVE LINEAR QUADRATIC REGULATOR (ILQR)

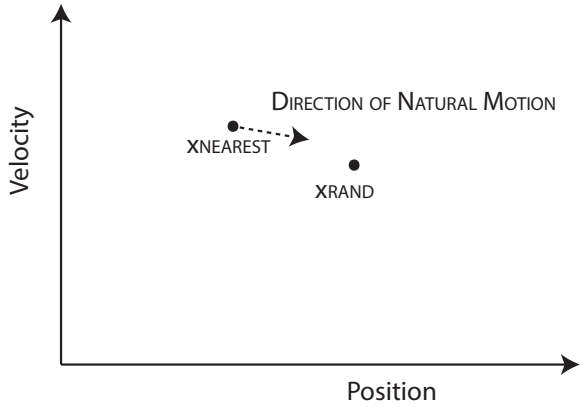


Figure 5.2: Influence of Linearization.

Typically, approaches in literature that aim to obtain an estimate of the distance metric in state space, use the concept of linearization of a nonlinear system around a point in the state space with zero velocity and/or zero inputs. This has been shown to work reasonably well in [44, 105]. However, these methods are only a good approximation of the nonlinear dynamics provided the point chosen for linearization is an equilibrium point. If this is not the case, the approximation is inaccurate.

This problem is illustrated in Figure 5.2, where, a typical scenario that arises during every iteration of an RRT. The direction of evolution of the natural system dynamics is

also shown. With a linearization approach such as in [105], the linearized model would be deprived of the velocity information at x_{rand} and hence, the cost-to-go to x_{rand} from x_{nearest} would yield a certain input value which eventually influences the cost. However, in reality, this input may not be needed or a very small input might be needed to steer the system to x_{rand}. Thus, approaches that use linearization around an equilibrium point could yield an inaccurate estimate of the actual cost-to-go.

As a consequence, approaches such as the LQR-RRT [105] would need a sufficiently large number of nodes before which a motion plan to the goal can be found. In case of simple nonlinear systems such as a pendulum, the effect might not be very evident. With more complex nonlinear dynamics such as in [83], the resulting increase in the number of nodes due to approximations inaccuracies leads to a significantly large amount of time for the LQR-RRT approach to converge to a solution.

In our work, we use the iLQR approach proposed in [83] for the approximation of the nonlinear dynamics. In this approach, linearization is carried out along a nominal trajectory instead of just a point in the state space. In doing so, a better approximation of the nonlinear dynamics can be achieved. The iLQR approach is presented here as a simple procedure in Algorithm 3. We limit the complete mathematical details due to space restrictions and those that are provided here are directly taken from [83].

Let us consider the discretized version of the system dynamics of Eq. 5.1:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{5.2}$$

with, the cost function is defined as,

$$\begin{aligned} J_0 = \frac{1}{2}\left(\mathbf{x}_N - \mathbf{x}^*\right)^T Q_f (\mathbf{x}_N - \mathbf{x}^*) + \\ \frac{1}{2}\sum_{k=0}^{N-1}\left(\mathbf{x}_k^T Q \mathbf{x}_k + \mathbf{u}_k^T R \mathbf{u}_k\right) \end{aligned} \tag{5.3}$$

where,

- $\mathbf{x}_N$ describes the final state after each execution of the input $\mathbf{u}_k$.
- $\mathbf{x}^*$ is the given target state.
- Q and Q_f are the state cost weighting matrices.
- R is the control-cost weighting matrix.

Using Eq. 5.2 and Eq. 5.3, the iLQR approach proceeds iteratively by obtaining a nominal open loop trajectory $\mathbf{x}_k$ by applying an input $\mathbf{u}_k$. With an initial input sequence $\mathbf{u}_k = 0$, each iteration produces an improved $\mathbf{u}_k$ by linearizing the system dynamics around the sequence $(\mathbf{x}_k, \mathbf{u}_k)$ and solving a modified LQR problem. The iterations continue until a cost convergence is achieved.

In Algorithm 3, the `ILQRCost` procedure, computes the cost of the nominal trajectory using Eq. 5.3. The `ILQRIterate` implements the linearization procedure mentioned earlier. Eventually, once the cost converges, the `ILQR` procedure returns the optimal cost-to-go between the states $\mathbf{x}_i$ and $\mathbf{x}_f$.

Algorithm 3 ILQR (uk, xi, xf, Qf, Q, R, dt, nIter)

$cost_{\text{curr}} \leftarrow \texttt{ILQRCost}\left(\mathbf{u}_k, \mathbf{x}_i, \mathbf{x}_f, Q_f, Q, R, dt\right)$
for $j = 1, \ldots n_{\text{Iter}}$ **do**
  $\delta\mathbf{u} \leftarrow \texttt{ILQRIterate}\left(\mathbf{u}_k, \mathbf{x}_i, \mathbf{x}_f, Q_f, Q, R, dt\right)$
  $\mathbf{u}_k^{'} \leftarrow \mathbf{u}_k + \alpha\,\delta\mathbf{u}_k$
  $cost_{\text{new}} \leftarrow \texttt{ILQRCost}\left(\mathbf{u}_k^{'}, \mathbf{x}_i, \mathbf{x}_f, Q_f, Q, R, dt\right)$
  if $cost_{\text{new}} - cost_{\text{curr}} < cost_{\text{threshold}}$ **then**
    `TerminateIteration`
  $cost_{\text{curr}} \leftarrow cost_{\text{new}}$
  $\mathbf{u}_k \leftarrow \mathbf{u}_k^{'}$
return $\left(\mathbf{u}_k^{opt}, cost^{opt}\right)$

Although iLQR is a reasonably good method to find the cost-to-go, it is not fast enough for use in planning. Therefore we will use Locally Weighted Projection Regression (LWPR) as a supervised learning technique to decrease the computation time of the cost-to-go, by initially incorporating a learning phase.

5.2.2. LOCALLY WEIGHTED PROJECTION REGRESSION (LWPR)

Locally Weighted Projection Regression (LWPR) is a supervised learning approach that has the potential to approximate nonlinear functions in high dimensional spaces [142]. It is a non-parametric and a fast learning approach. The nonlinear function is learned as a set of locally linear regression models along particular input dimensions. The linear models are eventually blended using a weighting function based on the area of validity of the local models to obtain the approximation of the nonlinear function.

Formally, given a data set in the form of input-output pairs $(\mathbf{x}, \mathbf{y})$, the LWPR method involves iterating the following function:

$$f(\mathbf{x}) = \frac{1}{W(\mathbf{x})} \sum_{k=1}^{K} \mathbf{w}_k(\mathbf{x}) \psi_k(\mathbf{x}) \tag{5.4}$$

where,

- $$W(\mathbf{x}) = \sum_{k=1}^{K} \mathbf{w}_k(\mathbf{x}) \tag{5.5}$$
- $\psi_k(\mathbf{x})$ are the local linear models.
- $\mathbf{w}_k(\mathbf{x})$ is a weighting factor depending on the area of validity of the local linear models.

It is particularly highlighted in [70] that the strength of LWPR learning lies in its ability to incrementally learn from training samples rather than learning from a collected set of samples. We however use the latter approach in this chapter. We undertake this approach only for simplicity and the incremental approach remains the main motivating factor for

our choice of LWPR learning over other supervised learning approaches such as Artificial Neural Networks. In fact, our eventual goal is to be able to learn the distance metric while constructing the RRT and this work is a first step in that direction.

In the following section, we combine the iLQR and the LWPR approaches that results in a novel method to approximate the distance metric function in the state space.

5.3. METHOD

In this section, we describe our novel approach in two parts. In the first part, we explain the creation of the training samples for the LWPR algorithm and the consequent learning of the distance metric function. In the second part, we explain the use of the learned distance metric function as a part of the RRT building process.

5.3.1. LEARNING THE OPTIMAL COST FUNCTION

Like for all supervised learning algorithms, providing an adequate amount of training samples is integral to LWPR learning to make a good approximation of the underlying function. A training sample consists of arguments to the function to be learned and the corresponding function value. In the context of our work, the function to be learned is the optimal cost-to-go between two states of a nonlinear dynamical system, given by the iLQR algorithm. The arguments to this function are two states in the state space. The procedure we use to learn the optimal cost function is shown in Algorithm 4.

Algorithm 4 LearnMetric (nSamples, (dt, nIter))

for $j = 1,\ldots\ n_{\text{Samples}}$ **do**
  $x_i \leftarrow$ `Sample`
  $x_f \leftarrow$ `Sample`
  $cost_{\text{optimal}} \leftarrow$ `ILQR` $(x_i, x_f, nIter, dt)$
  $X_{tr}(j) \leftarrow [x_i; x_f]$
  $Y_{tr}(j) \leftarrow cost_{\text{optimal}}$
return $\rho_{\text{learn}} =$ `LWPRlearnmetric` (X_{tr}, Y_{tr})

There are two main aspects of the `LearnMetric` procedure. The first is in the lines where the initial state x_i and desired final state x_f are sampled. In order to ensure an unbiased learning, these states are sampled randomly. Furthermore, the random sampling procedure also maintains the essence of the sampling used in RRT-based approaches where each incremental step involves a random sampling of a state from the state space.

The second aspect is in the last line of the procedure where the distance metric function ρ_{learn} is learned using the `LWPRlearnmetric` procedure. This is done by initially dividing the obtained samples into a training set and a test set. LWPR uses these two sets for evaluating the quality of the learned function.

5.3.2. USING THE LEARNED DISTANCE METRIC FOR RRT

Here, we describe the following step after learning the distance metric function. That is, using the learned distance metric ρ_{learn} while constructing the RRT. The procedure for RRT construction with the learned metric, `RRTLearn` is illustrated in Algorithm 5.

Algorithm 5 RRTLearn ((V, E), N)

for $j = 1,\ldots N$ **do**
  $x_{\text{rand}} \leftarrow$ `Sample`
  $x_{\text{nearest}} \leftarrow \rho_{\text{learn}}\,(V, x_{\text{rand}})$
  $[cost_{\text{xnew}}, x_{\text{new}}] \leftarrow$ `iLQR` $(x_{\text{nearest}}, x_{\text{rand}})$
  $X \leftarrow X \cup \{x_{\text{new}}\}$
  $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$
return $G = (V, E)$

We verify the correctness of the approximated function in two steps. First, the `iLQR` procedure attempts to connect the random state with the nearest state x_{nearest}. The resulting new state x_{new} and the corresponding optimal cost, $cost_{\text{xnew}}$ are compared to the predicted cost between x_{nearest} and x_{new} using ρ_{learn}. The cost prediction error, $cost_{\text{error}}$ is determined as follows:

$$cost_{\text{error}} = \left(\rho_{learn}(x_{\text{nearest}}, x_{\text{new}}) - cost_{\text{xnew}}\right)^2 \tag{5.6}$$

The performance of our algorithm is compared against the LQR-RRT approach in [105] in terms of the number of nodes required to find a solution. We now proceed by presenting the experimental results from our work.

5.4. EXPERIMENTAL RESULTS

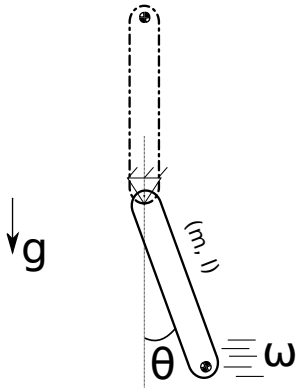


Figure 5.3: The pendulum setup considered for RRTLearn.

We consider the simple pendulum swing up problem (see Figure 5.3) to evaluate the RRTLearn algorithm. The physical parameters and the linearization approach are considered in accordance with [83]. We consider a pendulum with bob mass $m = 1\,\text{kg}$, length $l = 1\,\text{m}$ and a damping of $b_d = 0.01\,\text{kg}\,\text{s}^{-1}$ at the mounting joint. We assess the performance based on three factors; correctness of the learned metric, state space coverage and using the metric to perform planning. Our work has been implemented in MATLAB using the available resources for LWPR in [70] and for iLQR in [85]. For the RRT implementations, the Robotic Toolbox [24] has been used and adapted to our requirements.

We begin with the verification of the learned metric with the actual iLQR cost based on different number of training samples used for LWPR learning. We use Eq. 5.6 for this purpose.

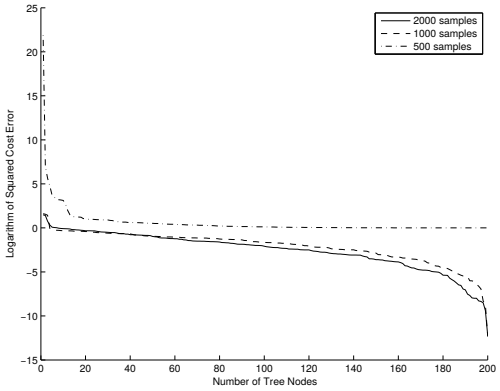


Figure 5.4: Logarithm of the squared cost error over all tree nodes.

It is clear from Figure 5.4 that the squared error is small for a majority of the nodes and those with large values are the inevitable outliers which are bound to be present when learning approaches are used. It is pertinent to note that the accuracy of the prediction drops considerably if the number of training samples is reduced.

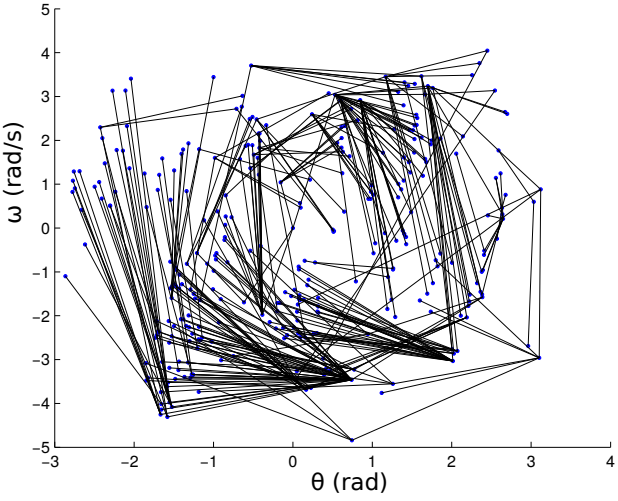


Figure 5.5: The edges of the tree are indicative of the natural evolution of the system dynamics of the pendulum. A similar observation is also identified in [44] to reason about the correctness of their distance metric approximation.

The correctness of learning is also further validated from the nature of the tree generated by RRTLearn. See Figure 5.5 for instance. The edges of the tree are indicative of the natural

evolution of the system dynamics. In other words, the neighbor selected for expansion is indeed the closest in state space to the randomly sampled state.

The main advantage of the RRTLearn approach is the speed-up achieved in computing the optimal cost-to-go between two states in the state space due to the use of learning. This is presented in Table 5.1 for different number of nodes considered for selecting the node to expand from. The resulting speedup from learning is evident from Table 5.1. It is important to note that as the number of nodes increase, the distance computation time using the learned approximation is higher in comparison to the LQR approach. This aspect is discussed in further detail in Section 5.5.

Table 5.1: Time for computing state space distance (Averaged over 20 runs).

	iLQR Metric	learned iLQR Metric	LQR Metric
1 Node	0.4127 s	0.000 16 s	0.0045 s
100 Nodes	41.18 s	0.0075 s	0.0057 s
500 Nodes	205.2 s	0.037 s	0.0081 s

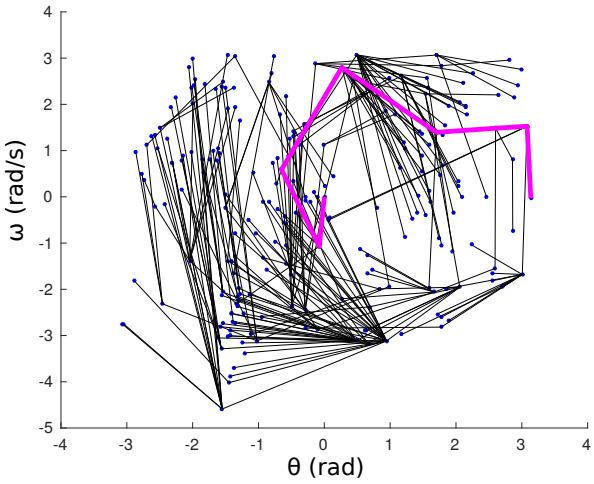


Figure 5.6: A candidate swingup solution found after a single run of RRTLearn.

We complement our earlier initial claim related to better state space coverage with our metric in comparison to existing approaches based on the number of tree nodes to reach a goal state. We believe that this is a reasonably good measure of state space coverage as it implies the existence of good node connectivity in the regions which represent the system dynamics in state space. This is different from the approach used in [44], where a discretization of the state space is used and number of nodes per discretized region is evaluated. We also compare our approach with the LQR-RRT approach. In all cases, we assume a goal region tolerance of $\left[0.1\,\text{rad}, 0.1\,\text{rad}\,\text{s}^{-1}\right]^T$. This is a region around the goal state, which when reached, indicates a solution has been found. In practice, the actual goal state would be reached by using a linear feedback controller once the goal region is reached. A candidate solution generated after one of the runs of RRTLearn is shown in Figure 5.6.

Table 5.2: Comparison of RRTLearn and LQR-RRT (Avg. Nodes).

	iLQR Steering	LQR Steering
Using RRTLearn	44.15 ± 9.52	167.05 ± 53.1
Using LQR-RRT	68.54 ± 16.05	208.7 ± 33.98

For evaluating the application of our metric to planning, we study four different cases based on the choice of the distance metric and the steering function. The results are presented in Table 5.2.

Table 5.3: Comparison of RRTLearn and LQR-RRT (Avg. Time).

	iLQR Steering	LQR Steering
Using RRTLearn	21.46 s	12.5 s
Using LQR-RRT	33.64 s	9 s

The first column of Table 5.2 indicates the distance metric used, and the first row indicates the choice of steering function. Each numerical entry in the table corresponds to a combination of the respective entry in the first column and first row and its associated 95 % confidence interval. The average number of nodes to find an initial solution[1] are computed over 20 runs of each choice. We note that the average number of nodes needed to find a plan to a goal position by RRTLearn using both iLQR and LQR steering is significantly less than the average number of nodes needed by the LQR metric ($p < 0.01$).

The LQR-RRT algorithm needs a higher average number of node evaluations to reach the goal region and this is a direct consequence of the linearization problem. However, it scores much better on the execution time (shown in Table 5.3) in comparison to RRTLearn for two reasons. In the simple case of pendulum, the linearization works well for most situations and hence the LQR metric is more or less reliable consequently allowing for reaching the goal region. Furthermore, the computation time of the LQR cost does not scale up with the number of tree nodes, which is the case with ρ_{learn} computation. However, it remains to be studied as to how this effect varies for systems with more pronounced nonlinear dynamical effects. Before we conclude, we present a brief discussion of our results and propose possible directions for future work.

5.5. DISCUSSION AND FUTURE WORK

From a kinodynamic planning perspective, we believe, RRTLearn with iLQR steering potentially yields better feasible plans compared to the LQR-RRT approach by virtue of the more appropriate linearization technique used. As a consequence, we expect RRTLearn to

[1] *Swing up* in our case is different to the common case considered, for example, in Reinforcement Learning problems where a policy consisting of back and forth motions is generated to achieve the desired goal. Our goal is to show that supervised learning can approximate the distance metric in state space and therefore enable RRTLearn to select the *most relevant* neighbor in the process of finding a solution.

provide better solutions over the LQR-based approaches for systems with stronger nonlinear dynamics. With our current results, we are limited to this benefit alone and further work is necessary to ascertain the possible benefits for online kinodynamic planning.

An inherent limitation with learning approaches such as the LWPR is the read out time increase with the number of input arguments. In the context of this work, as the number of nodes in the tree increases, the distance read out time also increases and as a consequence could nullify the speed-up achieved. However, by restricting the number of nodes to be evaluated, for example, by using efficient nearest neighbor search [147], this problem can be avoided. Another way to overcome this could be to use supervised learning techniques that are more efficient to read out, such as artificial neural networks. However, we plan to learn the distance metric while building the tree, focusing the learning in the most relevant states, and this requires an incremental method. Furthermore, in an online planning situation such as obstacle avoidance, a potential online kinodynamic solution could be to construct a new tree around the obstacle. In such a scenario, the number of nodes would certainly be a small number as the new tree would be rooted at the current state of the system. From an offline planning perspective though, our approach could potentially yield better plans (in terms of feasibility) because the nonlinear dynamics are better approximated by the iLQR approach in comparison to the standard LQR approach. Another benefit with the learning approach is that, the dynamical information is embedded in the learned function once during training and the same metric can be used multiple times, for instance, during re-planning.

As per our knowledge, our approach is one of the first in the literature that uses learning to approximate the optimal cost-to-go between states for RRT-based planning. In the future, we propose to evaluate the benefits of our approach on larger state-spaces by studying the feasibility of the generated plans and the amount of post processing needed to execute those plans on real robots. Additionally, constraints such as limits on the control inputs can be accounted for, while learning, by imposing cost penalties for constraint violations. An important aspect in our approach is the fact that the learned approximations are strongly dependent on the environment used for learning. However, the influence of discontinuities on the learning created by situations such as state limits or new obstacles remains to be studied in further detail.

5.6. CONCLUSIONS

Appropriate choice of distance metric for RRT-based approaches holds the key to successful planning in state space. Typically, this metric is the optimal cost-to-go between two states in the state space. However, obtaining this cost is computationally intensive for systems with nonlinear dynamics. In our work, we address this issue by decomposing the problem in two levels. We initially estimate the cost-to-go based on the Iterative Linear Quadratic Regulator (iLQR) principle proposed in [83] over a random state space distribution. This is followed by learning a nonlinear function that approximates this cost using a supervised learning technique called Locally Weighted Projection Regression (LWPR) [142]. We experimentally verify the correctness of the learned function and subsequently use this function to compute distances in state space while planning with RRT. Thus our approach provides the benefit of using a closer approximation of the optimal cost at high speeds. Our experiments have shown a speed-up in the distance metric computation of up to a factor of 1000.

6

RRT-CoLearn: towards kinodynamic planning without numerical trajectory optimization

When we ask advice, we are usually looking for an accomplice.

Joseph-Louis Lagrange

Sampling-based kinodynamic planners, such as Rapidly-exploring Random Trees (RRTs), pose two fundamental challenges: computing a reliable (pseudo-)metric for the distance between two random nodes, and computing a steering input to connect the nodes. The core of these challenges is a Two Point Boundary Value Problem, known to be NP-hard. In the previous chapter, we presented the idea of approximating the distance metric using supervised learning, reducing computation time drastically. However, the steering input was still computed online. Also, the principles of direct optimal control were used to generate the data for supervised learning. This chapter proposes to use indirect optimal control instead, because it provides two benefits: it reduces the computational effort to generate the data, and it provides a low dimensional parametrization of the action space. The latter allows us to learn both the distance metric and the steering input. This eliminates the need for a local planner in learning RRTs. Experimental results on a pendulum swing up show 10*-fold speed-up in both the offline data generation and the online planning time.*

The contents of this chapter have been derived from the paper:
Wouter Wolfslag, Mukunda Bharatheesha, Thomas Moerland and Martijn Wisse, *RRT-CoLearn: towards kinodynamic planning without numerical trajectory optimization*, IEEE Robotics and Automation Letters (RA-L), 2018.
The main difference to the paper is the inclusion of a section on the probabilistic completeness considerations of the proposed RRT-CoLearn algorithm.

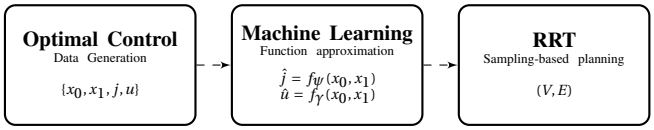


Figure 6.1: Schematic of the Learning-RRT architecture. First, optimal control generates a dataset that specifies the local steering cost j and control input u for a given start state x_0 and end state x_1, Subsequently, machine learning predicts the steering cost and input given a start and goal node. These first two (computationally intensive) phases happen *offline*. Subsequently, the RRT handles *online* planning. The function approximators provide the RRT with quick cost and input prediction without online optimization.

FOR motion planning of robotic manipulators, kinodynamic planning and sampling-based planning are getting increasingly popular. Kinodynamic planning, i.e., planning in state space rather than configuration space, improves robustness, speed and energy efficiency of robots [23, 146, 109]. Combining configuration-space planning with post-processing in state-space is often successful [108], but cannot solve all challenging dynamical problems. Sampling based planning has been shown to be the most viable way to handle high dimensional spaces and obstacles [51, 80]. Therefore, this chapter will consider how to apply Rapidly-exploring Random Trees (RRTs) [80], the most popular sampling-based single-query planning algorithm, directly to state space planning for deterministic, fully modeled systems.

RRT builds a tree graph structure with the states of the system as nodes and the trajectories of the system between two states as edges. The algorithm selects a node to expand from based on a *distance* to a randomly sampled node in the state space, i.e., it selects the *nearest* node in the current tree. That node is then expanded by means of a local planner that aims to reach the randomly-sampled node. These steps happen online, so computation time is crucial. Unfortunately, in state-space, both local planning and distance computation are computationally expensive [80].

In literature, the main approach to reduce the computational burden is to approximate the true distance function by a heuristic [44, 105, 63, 130]. Two frequently used heuristics are the Euclidean distance and the optimal steering cost for a linear approximation to the system. The convergence of RRT variants using the Euclidean distance heuristic, and random steering inputs, was extensively analyzed in [84]. For promising results for the linearizing heuristic see [47, 143, 126]. However, these heuristics only minimally utilize the dynamical properties of the system, and therefore typically require more nodes to solve a given problem using RRT.

Recent literature proposes a promising different approach, which we call Learning-RRT [12, 100, 3]. Learning-RRT involves an offline machine learning phase that learns the distance and steering function in RRT (Figure 6.1). An optimal control algorithm provides a database of optimal trajectories, which is the input for a supervised learning algorithm. This algorithm learns to approximate the functions the RRT requires. Note that trajectories found by RRTs are in general not optimal, even if the segments in the database are. Using optimal segments in the database provides a meaningful steering-cost metric and similarly shaped trajectories for connecting similar points in state space, which helps the learning algorithm. Supervised learning provides two benefits: 1) generalization over state-space and 2) fast online predictions. Thereby, the trajectory optimization does not have to be repeated for new situations, and is shifted offline.

In this chapter, we propose a method that helps to overcome the two remaining challenges of Learning-RRT:

1. **Local planning**: In literature on learning-RRT [12, 100], only the distance function is approximated by machine learning. Supervised learning of the steering function is hard due to the large number of parameters typically required to describe optimal input signals. Therefore, previous Learning-RRTs resort to computationally expensive optimizations for their steering function [12].

2. **Dataset generation**: The dataset for the supervised learning algorithm consists of many optimal trajectories. As optimizing a single trajectory already is a significant computational burden, generating a full dataset is very computationally demanding.

The main idea of this chapter is the use of indirect optimal control to generate the dataset. This allows us to solve both the problems mentioned above, thereby decreasing the computational burden by up to two orders of magnitude, both in the offline and the online phase of the learning RRT planning. However, the dataset generated by this method contains a bias, which is problematic for the learning algorithm. It turns out we can efficiently remove this bias through a simple dataset cleaning algorithm.

The focus of this chapter is the introduced method and its implementation details. Our experiments provide proof of concept by evaluating our method on a pendulum swing-up, as was done in Chapter 5. While simple, this task allows us to demonstrate and compare the validity of our method. To our knowledge, we are the first to demonstrate learning of the control input in a kinodynamic sampling-based planner.

This chapter is structured as follows. Section 6.1 describes the Learning-RRT algorithm. This algorithm is applicable to any data generation method, and intended to structure all Learning-RRT components. The subsequent sections combine to introduce RRT-CoLearn, which is a Learning-RRT. The main contribution comes in Section 6.2, which tackles the problems of *local planning* and *dataset generation* via indirect optimal control. The class of systems for which this approach is valid can be found in that section as well. Section 6.3 discusses how to remove the dataset bias introduced by optimal control. In Section 6.4, we discuss how the learned steering-cost metric affects the convergence of the RRT algorithm. Section 6.5 presents experimental validation of our algorithm. Finally, Sections 6.6 and 6.7 contain discussion and conclusions.

6.1. LEARNING-BASED RRT

Learning-based RRTs leverage (supervised) learning to speed up the computationally expensive modules of a kinodynamic RRT. The Learning-RRT algorithm is presented in Algorithm 6. Here we present the standard, forward search RRT version of the algorithm as used in the experiments. Enhancements are briefly discussed in Section 6.6.

The first step in the algorithm is to create a dataset of optimal trajectories through the state-space of the system ($\mathcal{X}$). This step disregards obstacles, as they can be handled at a later stage by an independent collision checker. The dataset $D = \{b^i\}_{i=1}^N$, has entries $b^i = \{x_0^i, x_1^i, j^i, u^i\}$ that consists of an initial state $x_0 \in \mathcal{X}$, a final state $x_1 \in \mathcal{X}$, a distance metric/local steering cost $j \in \mathbb{R}^+$, and a set of parameters $u \in \mathcal{U}$, that describe the optimal input leading the system from state x_0 to state x_1.

Algorithm 6 Learning RRT ((V, E), N)

$\hat{D} \leftarrow$ `generate_data`(N) // Section 6.2
$D \leftarrow$ `clean_data`$(\hat{D})$ // Section 6.3
$\hat{J} \leftarrow$ `fit_cost`(D)
$\hat{U} \leftarrow$ `fit_input`(D)
$\hat{V} \leftarrow$ `fit_valid`(D) // Section 6.4
$(X, E) \leftarrow (x_{\text{initial}}, \emptyset)$
solutionfound $\leftarrow$ False
while NOT(solutionfound) **do**
  $x_{\text{target}} \leftarrow$ `sample()`
  if $ANY(\hat{V}(x, x_{\text{target}}))\forall x \in X$ **then**
    $x_{\text{nearest}} \leftarrow \arg\min_{x \in X} \hat{J}(x, x_{\text{target}})$
    $(c, x, u) \leftarrow$ `simulate`$(x_{\text{nearest}}, \hat{U}(x_{\text{nearest}}, x_{\text{target}}))$
    if not `collision`(x_{nearest}, x) **then**
      $X \leftarrow X \cup \{x\}$
      $E \leftarrow E \cup \{(x_{\text{nearest}}, x, c)\}$
return X, E

Note that $\mathscr{U}$ can take many forms, depending on the discretization used. For example, it can be the coefficients of a polynomial or the values at the switch times of a piecewise-linear function.

The optimal trajectories are generally found using a numerical algorithm searching for local optima, which poses a challenge for the supervised learning algorithm. If two solutions are nearby in x_0 and x_1, but hail from a different local optimum, a deterministic supervised learning algorithm (for example trained on mean-squared error) will predict the average over the two solutions. This not only makes the cost prediction inaccurate, but most importantly it ruins the steering input prediction: the average of the two steering inputs in the data could lead to a completely different state than the target state. This local-optimum bias requires us to create the dataset D in two stages. The first stage creates a dataset $\hat{D}$ of size N which contains local-optimum-bias, and is indicated in Algorithm 6 by the function `generate_data`. The second stage `clean_data` removes the local-optimum-bias to create the desired dataset D.

The second step is to use a supervised learning algorithm on D to approximate the two functions that define the optimal control solution: 1. the function $\hat{J} : (\mathscr{X}, \mathscr{X}) \rightarrow \mathbb{R}^+$, which maps from an initial and a final state to a steering cost, 2. the function $\hat{U} : (\mathscr{X}, \mathscr{X}) \rightarrow \mathscr{U}$, mapping the initial and final state to the required input parameters.

For both function approximators we implement k-nearest neighbours [37], a standard non-parametric function approximator with robust performance in smaller state-spaces. For a test point x, we identify the k nearest neighbors in our dataset D. The predicted value (e.g. for cost j) for the test point is the average value of these neighbors. We cover extensions to other learning techniques in Section 6.6.

The next step, `fit_valid`, addresses an inherent limitation of supervised learning and is treated in Section 6.4.

The fourth stage of the Learning-RRT algorithm, the online RRT stage, handles long-term planning and obstacles. First, it samples a point and tests for a valid connection from

any node in the tree to the sampled point. If that exists, it expands the node that is nearest to the sampled point according to the function $\hat{f}$. The expansion will use the learned steering function $\hat{U}$, which likely makes a small error. The new node is therefore not exactly at the sampled state. The trajectory to the new state is checked for collisions. If collision free, the new node is added to the tree. The algorithm iterates these steps until it connects to the desired region in state-space. As standard in RRTs, the sampling of new nodes is biased towards the goal by intermittently replacing the sampled state with a desired end-state.

6.2. DATA GENERATION

The dataset for the function approximator is generated from a set of optimal trajectories. The most common approach to find these trajectories are the so-called *direct* optimal control approaches [111, 137]. In these approaches the state equations and cost function are approximated by a discretized system, which is then numerically optimized.

An alternative to direct optimal control is the much older *indirect* approach [117, 110], which first optimizes and then discretizes. For many applications, direct approaches replaced the indirect approach due to better numerical stability at long planning distances. However, it turns out indirect optimal control is ideally suited for the RRT scenario. First, the numerical instability poses no problem for the short segments that are required for RRT. Furthermore, indirect optimal control brings two important benefits. First, it parametrizes the control input in a low dimensional space, which allowing it to be learned. Second, it removes the need for optimization in the sampling process, which speeds up data generation. Both benefits are further explained at the end of this section, after the indirect optimal control method is described. At that point, we will also explain the remaining downside of the indirect optimal control approach: a more biased dataset.

INDIRECT OPTIMAL CONTROL

Here the indirect optimal control procedure is used to derive the optimal equations of motion for a pendulum swing-up. The procedure can be applied to other systems, with small differences. For more details and proofs see [99].

Indirect optimal control finds the functions $x(t)$ and $u(t)$ from time $t \in \mathbb{R}$ to state $x \in \mathbb{R}^n$ and input $u \in \mathbb{R}^m$, that minimizes a cost function of the following form:

$$J(x(t), u(t)) = \int_0^{t_\mathrm{f}} C(x(t), u(t)) \mathrm{d}t \tag{6.1}$$

$$\begin{aligned} \text{Subject to:} \quad & \dot{x}(t) = f(x(t), u(t)) \quad \forall\, t \in (0, t_\mathrm{f}), \\ & x(0) = x_\mathrm{initial}, \quad x(t_\mathrm{f}) = x_\mathrm{final} \end{aligned} \tag{6.2}$$

where x_initial and x_final are fixed initial and goal states, and the final time t_f is optimized along with $x(t)$ and $u(t)$. We will often drop the explicit dependency on the time t.

For the pendulum we get $f(x, u) = (\omega, \sin(\theta) + u)$, with $x = (\theta, \omega)$, θ and ω the (angular) position and velocity respectively, and u the torque. The cost integrand $C = 1 + u^2/2$, which takes into account both the time it takes to reach the goal-state as the control effort to do so.

The first step in the indirect optimal control approach is to define the Hamiltonian $\mathscr{H}$, the sum of the integrand C and the inner product of the state derivatives with a vector of

Lagrange multipliers, also called the costate. With $(\lambda_\theta, \lambda_\omega)$ as costate, we obtain:

$$\mathscr{H}(x, \lambda, u) = 1 + 1/2\, u^2 + \lambda_\theta \omega + \lambda_\omega(\sin(\theta) + u) \tag{6.3}$$

The second step is finding an optimal input u^*, by minimizing the Hamiltonian with respect to the input:

$$u^* = \arg\min_u \mathscr{H} = -\lambda_\omega \tag{6.4}$$

The third step takes partial derivatives of the optimal Hamiltonian, which is created by replacing the input with the optimal input: $\mathscr{H}^*(x, \lambda) = 1 + \lambda_\theta \omega + \lambda_\omega \sin(\theta) - 1/2 \lambda_\omega^2$. Note that this equation only depends on the state and costate. The partial derivatives form a system of ordinary differential equations (ODEs) specifying the evolution of the optimal state and costate over time:

$$\dot{\theta} = \frac{\partial \mathscr{H}^*}{\partial \lambda_\theta} = \omega \qquad \dot{\omega} = \frac{\partial \mathscr{H}^*}{\partial \lambda_\omega} = \sin(\theta) - \lambda_\omega \tag{6.5}$$

$$-\dot{\lambda}_\theta = \frac{\partial \mathscr{H}^*}{\partial \theta} = \lambda_\omega \cos(\theta) \qquad -\dot{\lambda}_\omega = \frac{\partial \mathscr{H}^*}{\partial \omega} = \lambda_\theta \tag{6.6}$$

The last step is to use the Eqs. 6.5-6.6 to find the optimal trajectory towards a desired state. For a given costate, the above equations are (numerically) integrated, which results in a locally optimal state trajectory. This trajectory depends on the choice of initial costate and the time duration of the integration. The initial costate and final time are tuned to find a locally optimal state trajectory that reaches the desired state. This tuning requires a numerical method that minimizes the difference between final state and desired state. Later we show that this tuning can be avoided when generating the database.

Optimal control problems solved for Learning-RRTs typically have a free final time and a cost integrand C that does not explicitly depend on time. On this type of problem, the constraint that sets the final time can be rewritten as a constraint on the initial costate [99]:

$$\mathscr{H}^*(x(0), \lambda(0)) = 0 \tag{6.7}$$

The main reason why *indirect* optimal control is largely replaced by direct methods, such as multiple shooting, is that the resulting differential equations are unstable, and therefore difficult to numerically solve reliably. However, because a learning-RRT database only requires short trajectories, this instability is not as important.

BENEFITS OF USING INDIRECT OPTIMAL CONTROL

There are two major advantages to using indirect optimal control for Learning-RRTs. First, the input directly follows from the costates; in principle, there is a map $U(x_{\text{initial}}, x_{\text{final}}) \rightarrow (\lambda_\theta(0), \lambda_\omega(0), t_f)$, from initial and final state to a set of only three parameters that describe the input function. This set is small, and will only grow linearly with the size of the state space (and is independent of the number of inputs). In contrast, direct optimal control approaches require to parametrize inputs as functions over time, which results in much larger spaces of parameters. The reduction in the number of parameters means the input function can be learned efficiently, thereby solving the problem of *local planning* in RRTs as identified in the introduction.

Algorithm 7 generate_data($N, T_{\text{final}}, r_{\text{bound}}$)

Optimal_ODEs ← Eqs. 6.1-6.6
$\hat{D}$ ← empty()
for $n = 1 : N$ **do**
  x_{initial} ← random_State()
  λ_{initial} ← random_Costate() s.t. Eq. 6.7
  T_{final} ← random_Time()
  x_{final}, J ← integrate(Optimal_ODEs,x_{initial} ,λ_{initial},T_{final})
  append($\hat{D}$,{x_{initial},x_{final},J,λ_{initial}})
return $\hat{D}$

To see the second major advantage, look at how the whole dataset is created. In current learning RRTs [12, 100], the dataset is generated by sampling from the $(x_{\text{initial}}, x_{\text{final}})$-space, and then, find an optimal trajectory and cost for each sample. Note that every combination of initial state, initial costate and final time produces an optimal trajectory for a certain final state. So, if we sample from the allowed initial states, initial costates and final times, we effectively sample over all initial state - final state combinations. Thus, we can eliminate the need for numerical optimization by sampling the costate, meaning the data are generated much faster.

The data generation procedure is outlined in Algorithm 7. The functions *random_State*, *random_Costate*, and *random_Time* sample random states, costates and final times depending on the problem domain and constraint Eq. 6.7. The function *integrate* numerically integrates the optimal control equations until the final time (T_{final}) or until the distance from the initial state reaches a reachability bound (r_{bound}). To optimize the efficiency of the data generation we add the intermediate integration results to the dataset, causing dependencies between the data-points. We observed that the learning algorithm performed well despite these dependencies.

In this section, we outlined the indirect optimal control approach to *data generation* in learning RRTs. Because the optimal control algorithm incorporates *costates*, we name the overall algorithm RRT-CoLearn. Its two major advantages are: learning optimal steering inputs (online speed) and generating data without optimizing (offline speed).

6.3. DATASET CLEANING

The dataset generated by Algorithm 7 originates from a search for local optima, and can therefore include a bias that interferes with learning performance. The problem is illustrated with an artificial dataset in Figure 6.2 (top), where in the middle input region we have the global optimum at the bottom, but there are local optima above it. A standard function approximator (with squared loss) for a given point input (independent variable) predicts the conditional expectation of the dependent variable, shown in green. Note that the predicted function deteriorates in the middle segment, where the average is predicted instead of the optimal cost.

This is problematic for predicting the cost function and especially harmful for predicting the control parameters. Averaging over two locally optimal control inputs by no means guarantees that we end up anywhere close to the target. As an intuitive example, consider

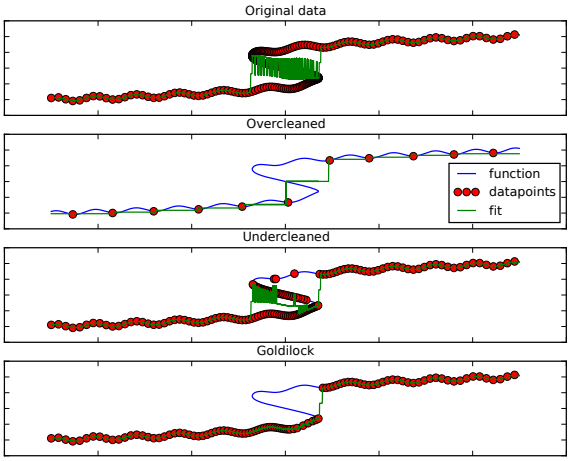


Figure 6.2: The data-bias problem and the effect of the d parameter in the data cleaning algorithm. The top figure shows an imaginary dataset, which has a problem with bias in the middle of its domain. The fitted function is a poor approximation of the least cost part of the datapoints. In the second figure d was chosen too large: the bias is gone, but there is not enough resolution left to accurately fit the function. In the third figure, d is too small: not all bias is removed. The bottom figure shows a proper choice for d: the bias is removed, and enough resolution remains.

the Dubins vehicle, which can be seen as a model of a non-holonomic car. The vehicle can reach a goal behind it by either steering left or right, but will fail to reach the target when taking the average (straight ahead). To counteract this issue, we need a dataset cleaning algorithm, i.e., a procedure that somehow eliminates conflicting (non-optimal) datapoints. In literature, there are resampling methods for dataset imbalance, most noteworthy class label imbalance in classification tasks [38]. However, our dataset is not imbalanced, but rather contains a systematic bias. It turns out we can leverage that structure to come up with a simple resampling/cleaning algorithm.

For each point in input space, we are interested in retaining the lower bound of the cost of the generated datapoints. First note that we prefer to remove points in high-density regions, as in low-density regions there is little to throw away, and we may only hope that our data are accurate. We implicitly remove from high density regions by first uniformly sampling a point from our dataset. We then search for its nearest neighbor in the dataset based on Euclidean distance. If this neighbor is within a distance d from our sampled point, we remove the node of the two with the highest cost, frequently removing a biased data-point while retaining a good one. Otherwise, we retain both points, which will happen in low density regions. This process is repeated until no points are removed for k_m consecutive steps, after which we return the cleaned dataset. The procedure is outlined in Algorithm 8.

The main parameter in this algorithm, d, can be interpreted as a neighborhood size. Figure 6.2 shows shows it has an optimal setting that depends on the dataset and which has to be found empirically. To evaluate it d, we fix it, run the cleaning, fit the function predictors, and then sample new datapoints to assess the error in cost and co-state parameters on the predictions. This evaluation is not ideal, but the ideal metric (RRT performance) is too expensive to compute.

Algorithm 8 clean_data($\hat{D}, d, k_{\max}$)

$D \leftarrow \hat{D}$
$k \leftarrow 0$
while $k < k_{\max}$ **do**
  $p_{\text{sample}} \leftarrow$ selectRandom(D)
  $p_{\text{neigh}} \leftarrow$ nearestNeighbor(p_{sample}, D)
  if distance(p_1, p_2) $< d$ **then**
    $k \leftarrow 0$
    $p_{\text{high}} \leftarrow \arg\min_{p \in \{p_{\text{sample}}, p_{\text{neigh}}\}} \text{Cost}(p)$
    remove(D, p_{high})
  else
    $k \leftarrow k+1$
return D

6.4. PROBABILISTIC COMPLETENESS CONSIDERATIONS

The machine learning approximations in RRT-CoLearn might intuitively interfere with its completeness properties. Therefore, we establish probabilistic completeness of our algorithm, by modifying the proof for the original RRT [80]. We have not tried to make the proof outlined here as general or rigorous as possible. The framework from [84] might aid in that effort, as might the ideas from [62].

We assume that there exists a solution to our planning problem which is built out of a finite number of shorter trajectories resulting from a finite set of inputs. That is, the solution has n waypoints $\mathcal{X} = \{x_0, x_1, \ldots, x_n\}$. The inputs between those waypoints are given by $\mathcal{U} = \{u_0, \ldots, u_{n-1}\}$. Later on, we will give one additional assumption on the set of possible inputs. We will show that given that this solution exists, RRT-CoLearn will eventually find it.

Consider that x_i is the most advanced waypoint of the solution that is already in the tree. The chance of reaching the next waypoint in the solution is factored as:

$$P(\text{reach } x_{i+1}) = P(u_i|\text{expand } x_i)P(\text{expand } x_i) \tag{6.8}$$

Now if we can guarantee both factors are strictly positive, i.e., $P(\text{expand } x_i) > 0$ and $P(u_i|\text{expand } x_i) > 0$, we get: $P(\text{reach } x_{i+1}) > 0$. This means the chance of getting to the next node of the solution is larger than zero, so at some point the algorithm will get to the next node, and the next one, and so on. Therefore the algorithm will eventually find the solution.

6.4.1. BOUNDING THE CHANCE OF PICKING THE RIGHT INPUT

The chance of picking the right input at the current waypoint, x_i is ensured if $P(u_i|\text{expand } x_i) > 0$. One way to do this relies on an additional assumption: that each u_i comes from a known finite set. As a consequence, each possible input can be given a non-zero probability. In our experiments, this finite set is constructed as a linearly spaced grid of initial costates with a stepsize of 0.01. We believe that a proof of completeness exists without this discreteness assumption, but we leave finding it for future work.

Note that our inputs are generated based on a prediction of the costate from the function approximator. If the input is constructed directly based on this prediction, the rest of

the inputs will have no chance of being selected which is not desirable. Therefore, a sufficient condition is to ensure that input resulting from the prediction is generated based on a probability distribution that associates a high probability of choosing a value close to the prediction, but also gives some probability for all other possible inputs. In our experiments, the control parameters are samples from truncated normals, with the bounds for each parameter specified by its sampled domain. The means are the value predicted by the learned model. The standard deviation σ of the (non-truncated)-normal is a parameter of the algorithm. The sampled inputs are projected on to the discrete input space.

Experiments showed that inaccurate prediction makes selecting the right input difficult when the problem requires the RRT to very precisely reach a small region in state space, as happens when near the goal region. Therefore, when the target in an RRT step involves the goal region, we increase the standard deviation of the input distribution. This empirically improved performance.

6.4.2. BOUNDING THE CHANCE OF PICKING THE RIGHT NODE

The chance of expanding from x_i, the most advanced of the set of solution nodes currently in the tree ($\mathcal{V}$), is the volume of the state space for which that node is the nearest node divided by the total volume of the free state-space:

$$P(x_i) = \frac{\text{Vol}(\{x \in \mathcal{X} | d(x_i, x) < d(x_e, x), \forall x_e \in \mathcal{V} \setminus x_i\})}{\text{Vol}(\mathcal{X})} \tag{6.9}$$

Note that, the entire state space $\mathcal{X}$ is considered in Eq. 6.9, instead of $\mathcal{X}_{\text{free}}$. This is important because of the following. The fact that x_i is the most advanced node in the solution implies $x_i \in \mathcal{X}_{\text{free}}$. In state space, this means two things: i) x_i is not a collision state and ii) x_i is also outside the region of inevitable collisions of any potential obstacles in the state space. The second part is crucial because it guarantees the existence of an ϵ-vicinity of x_i that always has some non-zero volume. As a consequence, there is always a chance of sampling some x that would still be in $\mathcal{X}_{\text{free}}$ and also nearest to x_i. The same will hold for all future nodes in the solution. In other words, our proof only holds under the condition that a different problem setting will still ensure the regions of inevitable collisions for the solution nodes remain intact.

Given this condition, we proceed further as follows. Since the volume of $\mathcal{X}$ is fixed and finite for a given sampling range and always larger than the volume encompassing x_i and its nearest neighbor, we only need to make sure the numerator is non-zero. This should be done while taking into account that the local steering cost function is piecewise continuous, with a discontinuity at 0.

The first step is to impose that the distance function must always be larger than some positive constant times the Euclidean distance:

$$d(x, y) \geq c_{\text{lb}} ||x - y||_2 \quad c_{\text{lb}} > 0 \quad \forall x, y \tag{6.10}$$

This lower bound ensures that the distance to a given node can not remain low over a large region of state space, causing the algorithm to always choose that node for expansion.

The distance should also have an upper bound for at least some volume of the state-space. We use the set $\mathcal{G}(x)$, the largest connected set containing x with points for which the distance

to x is bounded by c_{ub} times the Euclidean distance:

$$\mathscr{G}(x) = \text{argmax} \text{Vol}(\mathscr{S}) \text{ s.t. } \mathscr{S} \subseteq \{y | d(x,y) \leq c_{\text{ub}} ||x-y||_2\}, \\ x \in \mathscr{S}, \quad \mathscr{S} \text{ is connected}$$

$$\text{For the upper bound condition: } \text{Vol}(\mathscr{G}(x)) > c_{\text{v}} \; \forall x \tag{6.11}$$

Note that these conditions are not met in two frequently studied cases: 1. when the cost function is the integral of the squared input, 2. when the system is not small time locally accessible, as happens for instance in underactuated systems.

Take the largest ball $\mathscr{B}_\rho(x_i)$ centered around point x_i, such that $c_{\text{ub}}||x_i - y|| \leq c_{\text{lb}}||x-y||$ for all y in the ball, and all nodes x in the tree. Based on simple Euclidean geometry, $\rho > 0$. Also, by construction, the intersection $\mathscr{B}_\rho(x_i) \cap \mathscr{G}(x_i)$ has positive volume, and all points in that intersection are closer (by measure d) to node x_i than to any other node in the tree. Together this shows $P(\text{expand } x_i) > 0$.

If we had access to the true distance function, or an approximation of it that meets the conditions specified above, this would conclude the proof of convergence. However, learning algorithms are intended to generalize using interpolation, so may make large errors when extrapolating. This is especially true for Learning RRTs, for which this problem has not been identified in literature yet. In RRTs we *uniformly* sample state-space, while we have confined our dataset to only contain short motion segments. Therefore, sampled combinations (x_0, x_1) are often outside the dataset, where function approximation may make large errors, which might cause conditions 6.10 and 6.11 to be violated. Particularly, the approximated distance metric might greatly underestimate the steering cost from a certain node, causing that node to be incorrectly chosen for expansion.

This issue can be solved by a binary classifier that decides whether a query would yield a valid prediction, i.e., whether the dataset D covers the queried point. The implemented classifier $\hat{V} : (\mathscr{X}, \mathscr{X}) \to \text{v}$, with $v \in [\texttt{true}, \texttt{false}]$, simply computes the summed distance to the nearest neighbors of the queried point to the points in the dataset, and rejects the query point if this sum becomes too large. An alternative approach, explored in [130], relies on reachability: the notion that the dataset contains only short segments, meaning the final states should be reachable within a short period of time. To avoid violating conditions 6.10 and 6.11 by small approximation errors in the learned function, the predicted steering cost is clipped at $10^{\pm 5}$. This clipping has negligible effect on the computation.

6.5. EXPERIMENTS AND RESULTS

To test our approach, we perform experiments on the pendulum described in Section 6.2. The task is to move the pendulum from its stable equilibrium $(\theta, \omega) = (-\pi, 0)$ to its unstable equilibrium $(0, 0)$.

This experiment does not require obstacle handling; we do not explicitly focus on collision checking or obstacles in our work, as our focus is on the RRT itself and we assume the collision checking is handled by an external algorithm.

Data were generated and cleaned for separate epochs, with 300 runs of the RRT algorithm per epoch. The data for each epoch consist of 40 000 simulations, which ended when the local cost exceeds 2 or when the norm of the state difference with the initial state (r_{bound}) exceeds 1.5. Integration was done by the 4^{th} order Runge-Kutta algorithm with a time step of

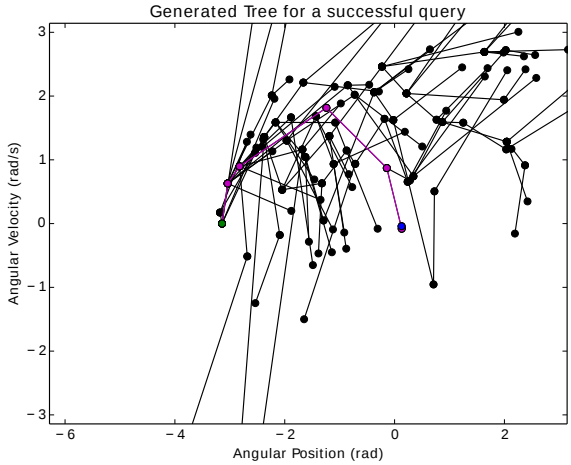


Figure 6.3: The state-space coordinates of the tree-nodes of a successful run of the algorithm. The edges are shown as straight lines, with those connecting the initial point to the goal being highlighted.

0.01 s. The initial position was uniformly sampled from $(-3\pi/2, \pi/2)$rad, the initial velocity from $(-\pi, \pi)$rad s^{-1}, and the initial costate sampled as described below. The data cleaning resolution d equals 0.05. The data cleaning stopping parameter $k_{\max}$ is set to 5000. The nearest neighbor fitting algorithm during the RRT takes $m = 3$ nearest neighbors. Finally, the standard deviation of the sampling distribution $\sigma = \pi/4$ normally, and $\pi/2$ when the query involves the goal state. The experiments ran on a MacBook with Intel(R) Core(TM) i5-3210M 2.50 GHz CPU and 8GB of RAM, running Ubuntu Linux 14.04 and code in Python, MATLAB and Julia.

To avoid projecting on the costate constraint (Eq. 6.7), which is computationally expensive for larger systems, the constraint is solved explicitly by uniformly sampling the parameter $\phi \in (-\pi/2, 3\pi/2)$ that sets the initial costate as follows: $\lambda_\theta = \tan(\phi)$ and $\lambda_\omega = -\sin(\theta) + \text{sign}(\cos(\phi))\sqrt{\sin(\theta)^2 + 2 + \tan(\phi)\omega}$. If λ_ω has an imaginary part, the simulation is disregarded.

Figure 6.3 shows a typical run of the algorithm. Three important points can be seen from this figure. First, the algorithm neatly expands through state-space. While it favours expanding along the circular paths that correspond to low input trajectories, it expands nodes in all feasible directions. This indicates that the distance metric works properly, and contrasts with trees that are grown without appropriate distance metric, which tend to have many nodes clustered. Secondly, the figure highlights the approximation errors that still exists; the edges that lead to the border of the figure all had target nodes inside the figure. As these nodes are outside the figure, the target was not reached exactly. Finally, the algorithm has tried to reach the goal-state a number of times from the same node. This shows the effect of the increased randomness; because the attempts are spread out sufficiently, the goal is eventually reached.

Figure 6.4 shows the computation times for 10 epochs. The planning time has the same variation in each epoch, indicating that data generation and cleaning are performed robustly. The median time to reach the target over all samples was 2.43 s, over 10 times faster than [12]

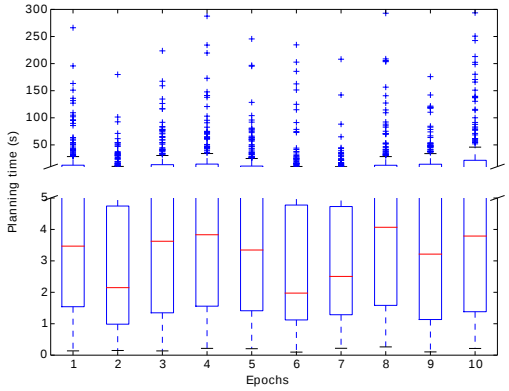


Figure 6.4: Boxplot of planning times for ten epochs.

on the same hardware. Data generation and cleaning took ~25 min per epoch, an order of magnitude faster than in [12][1]. The performance of the learned steering function is assessed by the mean squared error between the target state and the final state attained by using the predicted costate. The median of this error over all epochs is 0.11.

The quality of the combined machine learning is assessed using the number of nodes needed to reach the target. For the results in Figure 6.4, the median over all runs is 84 nodes. About 30 % smaller (better) than in [12], this result is best interpreted as a roughly equal performance, as the problem here does not require the pendulum to swing back and forth to reach its goal. Equal performance indicates the learned steering function approximates the online optimization well.

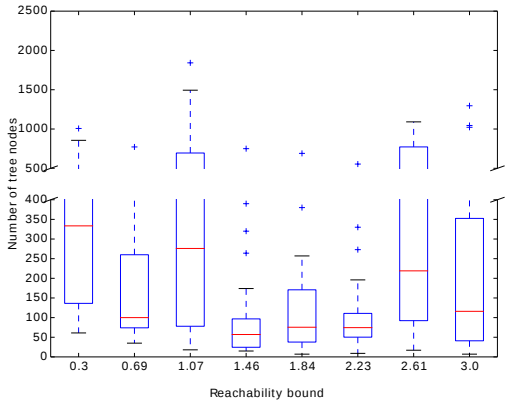


Figure 6.5: The number of nodes required to find a solution using different settings of the reachability bound.

Figure 6.5 shows the effect of the reachability bound on the number of nodes needed

[1] The cited paper does not report the offline computation time. However, due to overlapping authors we know offline computation took nearly a week.

to reach the goal position. This investigates how the length of the simulation in the dataset influences the performances of the algorithm. When the simulations are too short, the online algorithm is forced to use many segments, hindering convergence. When simulations are too long, the coverage of all possible trajectories becomes sparse, causing poorer learning performance.

Figure 6.6 shows the number of nodes needed to reach the goal position with various parts of the algorithm removed or replaced. This allows us to identify the important aspects of the algorithm, and to compare the algorithm to the state-of-the-art. The last four algorithms have the validity check turned off and perform worst. Note that [12] does not use such a check, but instead implements an online trajectory optimization to avoid the poor performance without validity check shown here. The figure also shows that the difference between learned and Euclidean distance is small when using a validity check, confirming the intuition from [130]. The algorithm in that paper is very close to the Δ__-settings in Figure 6.6. Finally, we see that learning the actions makes the algorithm perform better than using random costate selection.

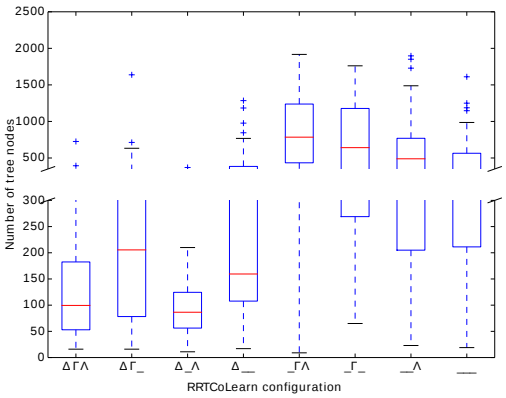


Figure 6.6: The number of tree nodes required to find a solution for different configuration settings for RRTCoLearn algorithm. The horizontal axis is labeled with three symbols, which can be turned on or off. In the first position, the Δ signifies the validity check being turned on. Similarly, the Γ switches between learned and Euclidean distance, and the Λ between learned and random costate (action) selection.

6.6. DISCUSSION

The algorithm introduced in this paper allows learning of not only the distance metric, but also the steering input. As proof of concept, we tested our algorithm on a basic pendulum swing up problem. It reduces the time spent both in the offline learning and in the online-computation by a factor of more than 10, a large step towards sampling based kinodynamic planning in a practical setting.

The main direction for future research is to support higher degree of freedom systems. Such systems will require more sample efficient data generation and learning. To improve the efficiency of data generation, new simulations might be selected based on already obtained

data, and the partially learned cost and steering functions, similar to what has been done for reinforcement learning [81]. (Deep) generative models could provide more efficient learning. These models allow sampling from complex, high-dimensional probability distributions, meaning we could retain all data points without averaging over solutions [46, 93].

A secondary direction for future research is to support input bounds. We have taken a preliminary step in this direction by implementing RRT CoLearn on a time optimal pendulum swing up with the torque bound set to 0.5, and all other constants as in Eq. 6.3. The resulting optimal equations of motion are invariant to the norm of the costate vector. Therefore the constraint of Eq. 6.7 is satisfied by setting the norm of the costate to 1. For 50 runs of CoLearn, the solution was always found within 2000 nodes, using a median planning time of 12.32 s, and median number of 195 nodes. These results compare favorably with those reported by the state-of-the-art in kinodynamic planning [108, 130].

Unfortunately, the input constraints can cause an exact overlap between trajectories starting from different costates, at least for a finite time. Such overlapping trajectories are difficult for the learning and cleaning algorithms we used. Therefore, a larger dataset was required, severely slowing down the RRT. Extending the machine learning aspects of CoLearn, such that they can cope with overlapping trajectories is an important theoretical and practical issue.

The third direction for future research is the combination of the CoLearn algorithm with other (non-learning) enhancements of the basic RRT algorithm, such as [72, 71]. RRT-Connect [72] would be the most prominent addition. It grows two trees: one forwards from the initial state, and one backwards from a goal state. A new model based on backwards integrated data must be learned for creating the backwards-searching tree. As the system of differential equations (Eqs. 6.5-6.6) is unstable in both forward and backward direction [99], the learning challenges remain similar.

Finally, note that RRTs, and therefore learning RRTs, are suited to finding novel motions for deterministic systems for which an accurate model is available. When planning for non-deterministic or uncertain systems, funnel based approaches are more appropriate than trajectories [138]. If the system has to perform similar motions repeatedly, multi query methods such as probabilistic roadmaps [41] or motion libraries can be more effective [8]. For the last problem, learning has already been used to find similarity between obstacle locations between planning instances [136]. It is interesting to investigate if using indirect optimal control similar to this chapter could be beneficial for those settings.

6.7. CONCLUSION

	Optimal Control	**Machine Learning**	**RRT**
+	Distance computation Optimal local planner	Generalization Fast online prediction	High dimensional Obstacle avoidance
-	Costly computation[6.2] Local optima→bias[6.3]	Needs large dataset[6.2] Needs unbiased data[6.3] Bad extrapolation[6.4]	Needs distance metric Needs local planner[6.2]

Table 6.1: Benefits and challenges of components of Learning-RRTs

This chapter first described a general Learning RRT algorithm. Table 6.1 shows the benefits and challenges of its components: Optimal Control, Machine Learning and RRT. The superscripts in the table refer to the sections in which the challenges are addressed.

RRT CoLearn, an instance of a learning-RRT, was proposed. CoLearn generates data faster and allows learning of the steering function, both by using indirect optimal control. It also uses a newly proposed data-cleaning algorithm for more accurate function approximation.

RRT CoLearn was successfully used on a pendulum swing up both with and without input constraints. The main results are on the system without input constraints and show that RRT CoLearn is 10 times faster than a state-of-the-art learning RRT [12].

NOTES

- The source code for RRT-CoLearn has been implemented using Python, MATLAB and Julia and is available at:
https://bitbucket.org/mbharatheesha/learning-optimal-control-for-rrt-matlab.
- A few topics concerning the scalability aspects of RRT-CoLearn to higher dimensional state spaces are presented in Appendix B.

THESIS CONCLUSIONS

7

Discussion and Conclusions

This thesis work presented a study into two relevant, yet contrasting aspects of robot motion planning. The first part focused on the use of sampling-based configuration space motion planners in industrial robotic applications and the second part focused on the challenges in sampling-based motion planning in state space. Before discussing the contributions of the two parts, we will briefly revisit the different challenges that were addressed in this thesis.

7.1. Motion planning in configuration space

Use of sampling-based motion planning algorithms in the context of industrial applications (bin picking) was the focus of the first part of this thesis. Here, we worked on two practical challenges associated with realizing a fully functional bin picking application.

7.1.1. Tuning of planning algorithm parameters

Software implementations of sampling-based algorithms in configuration space always involve one or more parameters that influence the success rate of planning queries. The primary function of these parameters is to limit the amount of programming required to adapt the implementation across different scenarios. However, manual tuning of these parameters for different scenarios is not desirable from a practical view point because it is cumbersome and more importantly, requires background knowledge of the algorithms to be able to extract the best performance from the planners. With a large diversity of planning algorithm implementations available, the tuning process only gets tedious. In this regard, we formulated the following research question:

How can we choose what is the best configuration space planner for a given bin picking scenario?

Chapter 2 presented an automatic framework to tune parameters of different configuration space planners in the Open Motion Planning Library (OMPL) for three different planning problems using industrial robots. The framework was built using an open source tool called Sequential Model based Algorithm Configuration (SMAC) [54]. The effect of the tuning

process on different planning algorithms was assessed using performance measures such as the number of planning queries successfully solved, the time required to find a solution and the path length of the resulting solutions. It was observed that for planners which had multiple parameters, the performance improved significantly. For example, the number of successfully solved planning queries doubled (~108 % improvement) for the BKPIECE planner which has 4 tuning parameters.

7.1.2. FUNCTIONAL SYSTEM INTEGRATION

Integrating a complete robotic solution for a bin picking application is a challenging task. This challenge is further compounded when the application changes from time to time such as in Small and Medium Scale Enterprises (SMEs) processing seasonal products. With bin picking tasks, the working environment of a robot is typically unstructured and often involves picking objects from a cluttered location. A motion planning module in such a scenario is required to quickly and reliably interact with other modules such as environment perception and act in a desired manner. While multiple open source software components addressing the motion planning requirements are available, realizing a fully functional industrial robotic system is still a complicated task. To this end, we formulated the following question:

What are the important challenges in integrating motion planning software components with other relevant components to realize a reliably functioning bin picking application?

Chapter 3 and Chapter 4 presented two cases of functional system integration. It was identified that robustness in performance, fast cycle times and reusability are the key challenges in the development of flexible and functional robotic solutions that can be used in SMEs. In Chapter 3 a software framework that used an off the shelf open source motion planning software as one of the components was presented. This motion planning component was subsequently fine tuned for maximizing robustness and minimizing cycle times for the bin picking tasks. Together with other fine tuned software components for environment perception, it was shown that a world championship winning bin picking robotic system for warehouse automation is achievable using current open source motion planning frameworks. The main contributor to the cycle times from a motion planning perspective was observed to be the collision checking process which ranged between $1-4$ s per object.

The motion planning module was also designed with a generic division of subtasks so that the same can be reused in a different application and thus minimizing the development and installation time. This proved highly effective in the initial development of the robotic system presented in Chapter 4. It was possible to reconfigure the motion module to meet the new application requirements in less than 3-days. However, the introduction and integration of the reactive collision avoidance behavior in the software framework was the most challenging task for this robotic system and approximately required about 6 weeks of software system integration work.

7.2. MOTION PLANNING IN STATE SPACE

Sampling-based motion planning in state space was the subject matter of the second part of this thesis. Here, we addressed two critical challenges that currently limit the possibility of using sampling-based motion planning methods in state space: computing the distance

metric and computing the steering input. In the following, the two challenges and our approaches are outlined.

7.2.1. DISTANCE METRIC APPROXIMATION IN STATE SPACE

The distance (pseudo) metric plays a crucial role in being able to successfully find a motion plan using a sampling-based planner in state space. The notion of *distance* in state space is the optimal cost-to-go between a pair of states. However, computing the optimal cost-to-go is time intensive in the context of (online) planning and leads to very large planning times to be used in practice. With a goal to eventually be able to utilize the benefits of sampling-based motion planning in state space, the following research question was formulated:

How can we alleviate the computational demands of the distance metric computation for motion planning in state space?

In Chapter 5, we proposed a supervised learning-based approach, *RRTLearn* to address this problem for the Rapidly exploring Random Tree (RRT) algorithm in state space. Our approach involves an offline data generation phase where an optimal control problem is solved between several randomly chosen state pairs to create a dataset containing state pairs and the corresponding cost. The iterative Linear Quadratic Regulator (iLQR) [83] is used as the optimal controller. The generated dataset is subsequently used in an offline supervised learning framework to learn an approximation of the cost-to-go across different regions in the state space. Locally Weighted Projection Regression (LWPR) [142] is used as the learning algorithm. Finally, the learned approximation is used in an online setting in the actual run of an RRT algorithm to solve a motion planning query. Our approach is verified on a simple pendulum swingup simulation problem. The proposed approach is shown to reliably approximate the state space distance metric with a speedup of about 3 orders of magnitude relative to solving one or more optimal control problems at each iteration of an RRT.

7.2.2. STEERING INPUT APPROXIMATION IN STATE SPACE

Approximating the optimal cost-to-go in a state space RRT contributes significantly to the reduction of the overall planning time. However, that reduction is still not enough when viewed from the perspective of using such methods in an online setting in practical applications. This is because, one optimal control problem per state pair still should be solved to generate a (possibly locally optimal) trajectory that connects the *nearest* state to the randomly sampled. Therefore, over an entire planning query, the solution times to these optimal control problems add up to the planning time. In pursuit of achieving further reductions in planning times in state space, the following research question was formulated:

How can we formulate the optimal control problem to quickly generate control inputs without compromising on the dynamical constraints?

The RRT-CoLearn algorithm is proposed in Chapter 6 as a promising approach to address this fundamental challenge. This algorithm is based on a combination of learning-based approximation and the principles of indirect optimal control. The key working idea of the RRT-CoLearn approach is that indirect optimal control formulation enables a parametrization of the control (or steering) input that is locally generalizable in different regions of state space,

as long as the desired trajectories are for a short duration. Similar to the aforementioned approach to approximate the distance metric, an offline data generation phase is utilized to create a dataset of state pairs, the corresponding costs and control input parameterization. Subsequently, the dataset is used to train a k-Nearest Neighbor (kNN) learning model to approximate the cost and the control input parameters. The learned model is eventually utilized in the actual run of the RRT algorithm to predict both the cost and control input parameters between the state pairs in an RRT iteration. Thus, the RRT-CoLearn approach eliminates the need to solve another optimal control problem to connect the nearest state to a randomly sampled state in an RRT. This approach is also tested on the problem of pendulum swingup in simulation. A further speed up of a factor of 10 is observed in the planning times relative to the previous experiment where only distance metric is approximated during the run of an RRT.

7.3. DISCUSSION

The rest of this chapter reflects on the different contributions from this thesis where the strengths and limitations of the proposed methods are highlighted. Subsequently, we present a few general conclusions and propose a few directions for further research.

7.3.1. FUNCTIONAL SYSTEM INTEGRATION WITH CONFIGURATION SPACE PLANNING

Planning in the configuration space of robotic manipulators is highly effective in solving motion planning problems where a robot needs to move between static start and end configurations. The principles of sampling-based planning have made it possible to generate fast planning solutions in high dimensional configuration spaces. For example, the planning solutions for the different bin picking scenarios presented in this thesis ranged between $(0.1 - 0.3\,\text{s})$ including the time parameterization process. The possibility of achieving such high speeds and the availability of open source software implementations make them attractive avenues for SMEs from an economical perspective. The first part of this thesis (Chapters 2 through 4) explored this potential in the context of industrial bin picking using the ROS-based MoveIt! [133] motion planning framework. Reliable and functional robotic systems were realized in the process.

One of the important observations from the first part of this thesis is that open source frameworks serve as a highly effective platform for quick modeling of the working environments and conduct simulation tests. This provides a tremendous advantage in making some preliminary design decisions on the robotic system hardware. For example, the ROS-Industrial Consortium[1] provides a large repository of various (software) models of industrial robots. In combination with motion planning frameworks such as MoveIt!, one can answer important design questions such as: Can a given robot perform the desired task effectively? Are additional components such as a rail or a second robot needed to meet certain cycle time requirements? All this is possible, *before* having to invest in actual robot hardware. This stands out as a key benefit for SMEs. On the flip side, an equally important aspect emerges from the perspective of *user-friendliness* of such frameworks for an industrial practitioner. In its current state, there is still a significant level of domain knowledge needed to build

[1] https://rosindustrial.org/

functional robotic systems. That is, a reasonable understanding of the planning algorithms and the software frameworks is required to reach the reliability levels desired from robotic systems at the industrial level. The framework presented in Chapter 2 attempts to minimize the requirement of motion planning domain knowledge to a certain extent by creating a *blackbox* algorithm tuning module. However, we are still quite far from abstracting these functionalities such that they can be integrated into the GUIs on teach pendants of industrial robots.

A second important observation is in the context of achieving reactive collision avoidance behaviors for bin picking applications in the context of SMEs. Collaborative robotic systems (also popular as Cobots) could serve as a strong alternative for SMEs processing seasonal products where setting up a full fledged production line is impractical. The central idea here is to create robotic systems that can co-exist with humans. This can be achieved by integrating basic sensory modalities with a robotic system such that information regarding its immediate vicinity is available. One such system is presented in Chapter 4 which was built to study to what extent one could seamlessly introduce a robotic system in an environment with humans. Two collision avoidance behaviors were realized when a human/obstacle gets too close to the robot (< 6 cm): i) stopping and continuing on the desired motion paths after there is sufficient clearance from the obstacle (ii) moving away from a potential collision by online modification of a motion path. Both behaviors have been successfully demonstrated at multiple public events to study their reliability in working environments with human presence. The latter behavior is desirable considering that there is no stopping involved which eventually influences the cycle times. However, the lack of global information (world model) is a key limitation of the system presented in Chapter 4. Some measures such as indirectly accounting for global information using constraints on the usable joint ranges of the robot are taken but they proved insufficient as there were situations where the robot would collide with its mounting while taking evasive actions. Yet another limitation of this system is the non-availability of redundant degrees of freedom to plan alternative motions around a desired motion path that is blocked by an obstacle. For example, there are only few alternatives to achieve/maintain a 3D end-effector pose (position and orientation) with a robot that has only 6 degrees of freedom. This further compounds the problem of finding alternate paths particularly when global information is unavailable. Therefore, it is desirable to have at least some form of redundancy so that reactive behaviors are effectively realized.

A final observation is regarding the benefits that the Robot Operating System (ROS) component based software framework offers in the context of robotic system integration. The results presented in Chapter 4 are achieved via collaborative software development across different research organizations. The component based software idea provided an elegant infrastructure for functional separation and interface standardization. ROS, being a middleware, provides an additional benefit by abstracting lower level software details such as message handling and communication with hardware. This enabled the possibility to focus solely on software behavioral and functional design aspects. However, one has to take extra care in using ROS-based components in applications with hard real-time requirements. For example, jerky robotic motions were observed when the software module communicating with the robot hardware was run in a non real-time mode and in the presence of other modules that had strong demands on processor times.

Thus a common observation from the first part of the thesis is that ROS-based open source software tools provide promising avenues for functional system integration for robotic bin picking. In the following a brief discussion on the second part of this thesis is presented.

7.3.2. SUPERVISED LEARNING FOR MOTION PLANNING IN STATE SPACE

The second part of this thesis focused on reducing planning times for sampling-based motion planning in state space. We proposed a framework using supervised learning tools to achieve this goal in the context of the RRT planner. In the process of designing these methods, multiple choices were made to develop this framework. Here, we present a discussion on these choices and their effects on the achieved results.

We started with the hypothesis that supervised learning is a potential tool to reliably approximate the optimal cost-to-go between a pair of states in state space. The main idea behind this hypothesis evolved as follows. Sampling-based planners provide a motion planning solution that is composed out of short trajectory segments between state pairs. Typically, the underlying dynamics in these trajectory segments exhibit locally linear properties. Further, similar state-pairs have approximately similar (locally) optimal costs between them. Therefore, it is likely that there exists a locally linear mapping between the *trajectory space* and the corresponding costs. Hence, we could possibly use supervised learning algorithms that are shown to be good at capturing and generalizing such (locally) linear relationships.

For supervised learning, an incremental learning algorithm, Locally Weighted Projection Regression (LWPR) was preferred over Artificial Neural Networks (ANN). The motivating factor for this choice was the possibility of incrementally improving the learning-based approximation model for the distance metric over each iteration of an RRT. However, there were two associated problems that hindered the utilization of the incremental learning capabilities. First, it was observed that at least 2000 state pairs were required to obtain a reliable approximation model for the 2-dimensional state space of the simple pendulum. In the days of big data, this number seems negligible. However, this translates to solving 2000 instances of an optimal control problem with each instance costing about 0.5 s of computation time. This means for the first run of a state space RRT where the learned model is not available, one has to wait at least 16 min before a learned model is available and probably a few seconds longer until a motion planning solution is found. This delay is perhaps still acceptable considering that the proposed offline training phase would exactly take the same time and subsequent runs of RRT will have faster planning times. However, the incremental approach with the RRT lead to a second, more important problem from a machine learning perspective. This problem is caused by the fact that the data used to learn the incremental models has a bias towards the nodes that are already in the RRT: one of the states in the state pairs used for computing the optimal costs will always be a part of the tree. This is a fundamental limitation as there is no guarantee that there is an adequate representation of data points from all regions of the state space. This typically leads to poor generalizations from the learned models. But, this is not the case with the offline training phase, as the state pairs are uniformly randomly sampled from the entire feasible range of the state space. Therefore it was decided to follow the approach as presented in Chapter 5.

The iLQR optimal control approach was used in Chapter 5 to compute the costs between state pairs. This choice was motivated by the requirement to better account for system dynamics while computing the (locally) optimal controller (and the associated costs) between

state pairs. However, the lack of a generalizable input parameterization in the iLQR approach meant that the dataset consisted of only state pairs and cost. As a result, solving one optimal control problem during the steering step of an RRT was still required and the speedup from learning was only limited to distance metric approximation. This limitation inspired the proposal to use the indirect optimal control approach in Chapter 6 where a generalizable control input formulation was possible using the concept of costates. The resulting approach, RRT-CoLearn, was evaluated on the simple pendulum swingup problem. However, it was observed that this approach does not yet directly scale well to higher dimensional state spaces. This is because, the idea of similar states leading to similar costs does not directly extend when control inputs are considered. Typically, the cost is a strictly positive real number. Therefore, commonly used machine learning approximations such as weighted averaging or interpolation during the prediction process provide a meaningful approximation of the real cost [11]. However, the same does not hold for control inputs as they are not necessarily strictly positive real numbers. As a result *similar* datapoints strongly *interfere* with each other and an averaging or interpolation operation would simply produce incorrect approximations. The proposed data cleaning process in Chapter 6 is a simple way to eliminate such interfering data points. However, it used a heuristic to identify and remove similar datapoints. The tuning of this heuristic is non-intuitive when the state dimensionality increases and could become a tedious process to arrive at an appropriate value.

Another fundamental challenge was observed while performing preliminary studies in extending the RRT-CoLearn to higher dimensional state spaces. The choice of the combination of costates that satisfy the initial conditions on the optimal hamiltonian (see Eq. 6.7 in Chapter 6) significantly impact the coverage of the different regions of the state space. For the two dimensional state space with a Lagrangian formulation of the system dynamics, the initial condition was a relatively simple quadratic equation in the two costates. An assumption on one of the costates was possible and hence created the possibility to uniformly cover the entire range of the other costate to have a good coverage of datapoints for the learning process. However, this simplicity does not prevail as the dimensionality increases. To begin with, the initial condition becomes a higher order polynomial (the polynomial order is the same as the dimensionality of the state space) in the costates. With only one equation in multiple variables, it is difficult to guarantee that the complete range for all the costates is covered. Secondly, the complexity of the equation increases making it difficult to use factorization techniques to uniformly cover the solution space of the equation. In summary, the RRTCoLearn seems to be a very promising direction towards making the next step towards realizing planning times in state space motion planning that are practical to use. However, there are some important fundamental problems that need to be addressed to realize its full potential. A few potential directions to address these problems with some preliminary results are presented in Appendix B.

Sampling-based motion planning in state space is an active topic of research. The results from this thesis and also other relevant results such as in [3] point towards promising avenues that can be explored further to eventually realize state space motion planning in practice. However, achieving planning times equivalent to or faster than what is possible in the configuration space is still a significant amount of work.

7.4. CONCLUSIONS

Based on the different aspects discussed before and the results from the different chapters, the main conclusions of this thesis are:

- Off the shelf open source software components for configuration space motion planning can definitely be configured quickly in the context of flexible robotic system integration for SMEs. However, further effort is necessary to make a full step towards use in industry from the perspective of user-experience and software architecture standardization.
- Supervised learning can successfully approximate the cost-to-go between state pairs in a state space without discontinuities. As a consequence, a significant speed up in the nearest neighbor determination phase of sampling-based algorithms can be achieved.
- Supervised learning in combination with indirect optimal control and adequate data processing can also approximate the steering input to connect state pairs in a given state space. Thus, a further speed up results in the total planning time for sampling-based planning algorithms in state space.
- In general, a combination of the principles of supervised learning and optimal control theory provides interesting and promising avenues to explore the domain of sampling-based motion planning in state space

7.5. FURTHER RESEARCH DIRECTIONS

This thesis presented a study into the field of sampling-based motion planning in configuration and state spaces. Industrial and academic systems were considered as examples to establish different results. Here, we present some pointers for further research.

A specific focus on software architecture design for typical industrial applications such as bin picking could help bridging the current gap between academia and industry in using off the shelf open source software components. In particular, inclusion of aspects related to robust and reliable performance such as failure mode handling, basic and advanced behavior design and synthesis can be interesting avenues to explore in the context of collaborative robot applications. Personnel in industries are typically skeptical of open source software components as there is no explicit software quality assessment or maintenance from a qualified company or an individual such as from a standards organization. The ROS-Industrial consortium and the recently funded European Union Project ROSIN[2] are aimed at taking steps in these directions.

The methods proposed in this thesis in the context of motion planning in state space were focused on establishing the proof of concept of the use of supervised learning tools to speed up the generation of planning solutions. While the concept has been shown to work, several interesting avenues could be explored such as the use of faster and sophisticated learning tools that can deal with the data bias problem. One such avenue could be the utilization of deep generative models such as the Conditional Variational Autoencoder (CVAE) [46, 94].

Short trajectory segments with a fixed and finite simulation duration were used in this thesis to generate the data points for learning. This works well when control input constraints

[2] www.rosin-project.eu

are not considered or if the system is underactuated. Therefore it would be interesting to investigate the effect of introducing input constraints on the proposed methods. Perhaps a potential further research direction is to use variable times for trajectory generation and also incorporate this information into the learning process (see [97]). In the context of input constrained systems, learning this information could be useful to generate planning solutions with *energy pumping* motions. In general, accounting for input bounds is especially challenging in the context of using learning-based predictions. Extra care has to be exercised to account for the effect of extrapolation where the predicted control inputs could potentially exceed the desired bounds. This is also an interesting prospect for further research.

REFERENCES

[1] E. Ackerman. *German Warehouse Robots Tackle Picking Tasks*. 2016. URL: https://spectrum.ieee.org/automaton/robotics/industrial-robots/german-warehouse-robots-tackle-picking-tasks (Accessed on: 10 June 2018).

[2] E. Aertbeliën and J. De Schutter. 'eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2014, pp. 1540–1546.

[3] R. Allen and M.Pavone. 'A Real-Time Framework for Kinodynamic Planning with Application to Quadrotor Obstacle Avoidance'. In: *AIAA Guidance, Navigation, and Control Conference*. 2016, pp. 5021–5028.

[4] *Amazon Robotics Challenge 2016*. URL: https://www.amazonrobotics.com/#/roboticschallenge/past-challenges (Accessed on: 9 September 2016).

[5] *Amazon Robotics Challenge 2017*. URL: https://www.amazonrobotics.com/#/roboticschallenge (Accessed on: 30 May 2017).

[6] V. I. Arnol'd. *Mathematical Methods of Classical Mechanics*. 2nd Edition. Springer, 1989.

[7] J. 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)*. 1990, pp. 1712–1717.

[8] D. Berenson, P. Abbeel and K. Goldberg. 'A robot path planning framework that learns from experience'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2012, pp. 3671 –3678.

[9] F. Berkenkamp, A. Krause and A. Schoellig. 'Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics'. In: *arXiv preprint arXiv:1602.04450* (2016).

[10] P. J. Besl and N. D. McKay. 'A method for registration of 3-D shapes'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256.

[11] M. Bharatheesha and M. Wisse. 'Supervised Learning: A potential tool for motion planning in state-space.' In: *Workshop on Beyond Geometric Constraints: Planning for Solving Complex Tasks, Reducing Uncertainty, and Generating Informative Paths and Policies*. 2015.

[12] M. Bharatheesha et al. 'Distance metric approximation for state-space RRTs using supervised learning'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2014, pp. 252–257.

[13] M. Birattari et al. 'F-Race and iterated F-Race: An overview'. In: *Experimental methods for the analysis of optimization algorithms (Chapter 13)* (2010).

[14] J. E. Bobrow, S. Dubowsky and J. S. Gibson. 'Time-optimal control of robotic manipulators along specified paths'. In: *The International Journal of Robotics Research* 4.3 (1985), pp. 3–17.

[15] R. Bohlin and L. E. Kavraki. 'Path planning using lazy PRM'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2000, pp. 521–528.

[16] R.A. Brooks. 'Intelligence without representation'. In: *Artificial intelligence* 47.1-3 (1991), pp. 139–159.

[17] E. F. Camacho and C. B. Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[18] B. Chazelle. *Approximation and Decomposition of Shapes.* Princeton University Press, 1987.

[19] P. Cheng. 'Sampling-based motion planning with differential Constraints'. PhD thesis. University of Illinois at Urbana-Champaign, 2005.

[20] P. Cheng and S. M. LaValle. 'Reducing metric sensitivity in randomized trajectory design'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2001, pp. 43–48.

[21] P. Cheng, G. Pappas and V. Kumar. 'Decidability of Motion Planning with Differential Constraints'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2007, pp. 1826–1831.

[22] B. Cohen and M. Likhachev. *The Search Based Planning Library (SBPL)*. 2009. URL: http://www.ros.org/wiki/sbpl (Accessed on: 10 June 2018).

[23] S. H. Collins et al. 'Efficienct bipedal robots based on passive-dynamic walkers'. In: *Science* 307 (2005), pp. 1082–1085.

[24] P. I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[25] N. Correll et al. 'Analysis and Observations From the First Amazon Picking Challenge'. In: *IEEE Transactions on Automation Science and Engineering* 15.1 (2017), pp. 1–17.

[26] A. Weiss D. Fischinger and M. Vincze. 'Learning grasps with topographic features'. In: *The International Journal of Robotics Research* 34.9 (2015), pp. 1167–1194.

[27] E. Dean-Leon et al. 'From multi-modal tactile signals to a compliant control'. In: *IEEE-RAS 16th International Conference on Humanoid Robots*. 2016, pp. 892–898.

[28] E. Dean-Leon et al. 'Robotic technologies for fast deployment of industrial robot systems'. In: *42nd Annual Conference of the IEEE Industrial Electronics Society (IECON)*. 2016, pp. 6900–6907.

[29] E. Dean-Leon et al. 'TOMM: Tactile Omnidirectional Mobile Manipulator'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 2441–2447.

[30] R. Diankov. 'Automated construction of robotic manipulation programs'. PhD thesis. Carnegie Mellon University, Robotics Institute, 2010.

[31] I. Dianov et al. 'Extracting general task structures to accelerate the learning of new tasks'. In: *IEEE-RAS International Conference on Humanoid Robots*. 2016, pp. 802–807.

[32] E. W. Dijkstra. 'A note on two problems in connexion with graphs.' In: *Numerische Mathematik* (1959).

[33] B. Donald et al. 'Kinodynamic Motion Planning'. In: *Journal of the Association for Computing Machinery* 40.5 (1993), pp. 1048–1066.

[34] C. Eppner et al. 'Lessons from the Amazon Picking Challenge: Four Aspects of Building Robotic Systems'. In: *Robotics: Science and Systems XII*. 2016.

[35] G. Francesca et al. 'AutoMoDe: A novel approach to the automatic design of control software for robot swarms'. In: *Swarm Intelligence* 8.2 (2014), pp. 89–112.

[36] B. Friedland. *Control System Design: An Introduction to State Space Methods*. Dover Publications Incorporated, 2005.

[37] J. Friedman, T. Hastie and R. Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin, 2001.

[38] M. Galar et al. 'A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches'. In: *IEEE Transactions on Systems, Man, and Cybernetics* 42.4 (2012), pp. 463–484.

[39] J. D. Gammell, S. S. Srinivasa and T. D. Barfoot. 'Batch Informed Trees (BIT*): Sampling-based optimal motion planning via the heuristically guided search of implicit random geometric graphs'. In: *International Conference on Robotics and Automation (ICRA)*. 2015, pp. 3067–3074.

[40] J. D. Gammell, S. S. Srinivasa and T. D. Barfoot. 'Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic'. In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*. 2014, pp. 2997–3004.

[41] R. Geraerts and M. H. Overmars. 'A comparative study of probabilistic roadmap planners'. In: *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 43–57.

[42] R. Girshick. 'Fast r-cnn'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1440–1448.

[43] R. Girshick et al. 'Rich feature hierarchies for accurate object detection and semantic segmentation'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[44] E. Glassman and R. Tedrake. 'A Quadratic Regulator-Based Heuristic for Rapidly Exploring State Space'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 5021–5028.

[45] X. Glorot and Y. Bengio. 'Understanding the difficulty of training deep feedforward neural networks.' In: *Aistats*. Vol. 9. 2010, pp. 249–256.

[46] I. Goodfellow. 'NIPS 2016 tutorial: Generative adversarial networks'. In: *arXiv preprint arXiv:1701.00160* (2016).

[47] G. Goretkin et al. 'Optimal sampling-based planning for linear-quadratic kinodynamic systems'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 2429–2436.

[48] P. E. Hart, N. J. Nilsson and B. Raphael. 'A formal basis for the heuristic determination of minimum cost paths.' In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[49] K. Hauser. 'Lazy Collision Checking in Asymptotically-Optimal Motion Planning'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2951–2957.

[50] C. Hernandez et al. 'Team Delft's Robot Winner of the Amazon Picking Challenge 2016'. In: *CoRR* abs/1610.05514 (2016).

[51] D. Hsu et al. 'Randomized Kinodynamic Motion Planning with Moving Obstacles'. In: *The International Journal of Robotics Research* 21.3 (2002), pp. 233–255.

[52] F. Hutter. 'Automated configuration of algorithms for solving hard computational problems'. PhD thesis. University of British Columbia, 2009.

[53] F. Hutter, H. Hoos and K. Leyton-Brown. 'An evaluation of sequential model-based optimization for expensive blackbox functions'. In: *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*. 2013.

[54] F. Hutter, H. Hoos and K. Leyton-Brown. 'Sequential model-based optimization for general algorithm configuration'. In: *Learning and Intelligent Optimization* (2011), pp. 507–523.

[55] F. Hutter et al. 'ParamILS: An automatic algorithm configuration framework'. In: *Journal of Artificial Intelligence Research* 36.1 (2009), pp. 267–306.

[56] L. Jaillet, J. Cortés and T. Siméon. 'Sampling-based path planning on configuration-space costmaps'. In: *IEEE Transactions on Robotics* 26.4 (2010), pp. 635–646.

[57] L. Jaillet et al. 'EG-RRT: Environment-Guided Random Trees for Kinodynamic Motion Planning with Uncertainty and Obstacles'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011, pp. 2646 –2652.

[58] L. Janson et al. 'Fast Marching Tree: A fast marching sampling-based method for optimal motion planning in many dimensions.' In: *The International Journal of Robotics Research* 34.7 (2015), pp. 883–921.

[59] Jörg-Rüdiger and Jorge Urrutia. *Handbook of Computational Geometry*. Elsevier, 1999.

[60] K. Ramirez-Amaro et al. 'General Recognition Models Capable of Integrating Multiple Sensors for Different Domains'. In: *IEEE-RAS International Conference on Humanoid Robots*. 2016, pp. 306 –311.

[61] M. Kalakrishnan et al. 'STOMP: Stochastic trajectory optimization for motion planning'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 4569–4574.

[62] S. Karaman and E. Frazzoli. 'Sampling-based algorithms for optimal motion planning'. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894.

[63] S. Karaman and E. Frazzoli. 'Sampling-Based Optimal Motion Planning for Nonholonomic Dynamical Systems'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 5041–5047.

[64] R. Katzschmann et al. 'Towards online trajectory generation considering robot dynamics and torque limits'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013, pp. 5644 –5651.

[65] L. E. Kavraki et al. 'Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces'. In: *IEEE Transactions on Robotics and Automation* 12 (1996), pp. 566–580.

[66] O. Khatib. 'The Potential Field Approach and Opertational Space Formulation in Robot Control'. In: *Adaptive and Learning Systems* (1986), pp. 367–377.

[67] N. Killingsworth, J. Nick and M. Krstić. 'PID tuning using extremum seeking: online, model-free performance optimization'. In: *Control Systems* 26.1 (2006), pp. 70–79.

[68] J. Kim, J. Keller and V. Kumar. 'Design and verification of controllers for airships'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2003, pp. 54 –60.

[69] J. King and M. Likhachev. 'Efficient Cost Computation in Cost Map Planning for Non-Circular Robots'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2009, pp. 3924 –3930.

[70] S. Klanke, S. Vijayakumar and S. Schaal. 'A library for Locally Weighted Projection Regression'. In: *Journal of Machine Learning Research* 9 (2008), pp. 623–626.

[71] R. A. Knepper, S. S. Srinivasa and M. T. Mason. 'Toward a deeper understanding of motion alternatives via an equivalence relation on local paths'. In: *International Journal of Robotics Research* 31.2 (2012), pp. 167–186.

[72] J. J. Kuffner-Jr and S. M. LaValle. 'RRT-Connect: An Efficient Approach to Single-Query Path Planning'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2000, pp. 995 –1001.

[73] J-C. Latombe. *Robot Motion Planning*. Kluver Academic Publishers, 1991.

[74] J-P. Laumond. 'Kineo CAM: A success story of motion planning algorithms'. In: *IEEE Robotics and Automation Magazine* 13.2 (2006), pp. 90–93.

[75] J-P. Laumond, N. Mansard and J-B. Lasserre. 'Optimality in Robot Motion: Optimal versus Optimized Motion'. In: *Communications of the Association for Computing Machinery (ACM)* 57.9 (2014), pp. 82–89.

[76] J-P. Laumond, N. Mansard and J-B. Lasserre. 'Optimization as Motion Selection Principle in Robot Action'. In: *Communications of the Association for Computing Machinery (ACM)* 58.5 (2015), pp. 64–74.

[77] S. M. LaValle. *From dynamic programming to RRTs: Algorithmic design of feasible trajectories*. Springer Tracts in Advanced Robotics, 2003, pp. 19–37.

[78] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[79] S. M. LaValle. *Rapidly Exploring Random Trees - A New Tool for Path Planning*. Tech. rep. Department of Computer Science, Iowa State University, 1998.

[80] S. M. LaValle and J. J. Kuffner-Jr. 'Randomized Kinodynamic Planning'. In: *The International Journal of Robotics Research* 20 (2001), pp. 378–400.

[81] S. Levine and V. Koltun. 'Guided Policy Search'. In: *30th International Conference on Machine Learning*. 2013, pp. 1–9.

[82] S. Levine et al. 'Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection'. In: *CoRR* abs/1603.02199 (2016).

[83] W. W. Li and E. Todorov. 'Iterative linear quadratic regulator design for nonlinear biological movement systems'. In: *International Conference on Informatics in Control, Automation and Robotics*. 2004, pp. 222–229.

[84] Y. Li, Z. Littlefield and K. Bekris. 'Asymptotically optimal sampling-based kinodynamic planning'. In: *International Journal of Robotics Research* 35.5 (2016), pp. 528–564.

[85] T. Lillicrap. *A MATLAB Implementation of the Iterative Linear Quadratic Regulator (iLQR)*. Google DeepMind. URL: http://contrastiveconvergence.net/~timothylillicrap/projects.php (Accessed on: 10 June 2018).

[86] T. Lozano-Perez. 'Spatial planning: A configuration space approach.' In: *IEEE Transactions on Computing* C-32.2 (1983), pp. 108–120.

[87] M. Lüdtke. *Integration Platform and Deployment Environment*. 2017. URL: https://github.com/ipa-mdl/ipde_utils (Accessed on: 10 June 2018).

[88] N. Mansard et al. 'A versatile Generalized Inverted Kinematics implementation for collaborative working humanoid robots: The Stack Of Tasks'. In: *International Conference on Advanced Robotics*. 2009, pp. 1–6.

[89] N. Mellado, D. Aiger and N. J. Mitra. 'Super 4PCS Fast Global Pointcloud Registration via Smart Indexing'. In: *Computer Graphics Forum* 33.5 (2014), pp. 205–215.

[90] D. Mellinger and V. Kumar. 'Minimum snap trajectory generation for quadrotors'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 2520–2525.

[91] P. Mittendorfer and G. Cheng. '3D surface reconstruction for robotic body parts with artificial skins'. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 4505–4510.

[92] P. Mittendorfer, E. Yoshida and G. Cheng. 'Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot.' In: *Advanced Robotics* 29.1 (2015), pp. 51–67.

[93] T. M. Moerland, J. Broekens and C. M. Jonker. 'Learning Multimodal Transition Dynamics for Model-Based Reinforcement Learning'. In: *arXiv preprint arXiv:1705.00470* (2017).

[94] T. M. Moerland, W. J. Wolfslag and M. Bharatheesha. *A dataset bias problem for learning-RRT, with two potential solutions*. Delft University of Technology. 2017. URL: http://thomasmoerland.nl/wp-content/uploads/2017/10/2017_DWRL_BharatheeshaWolfslagMoerland.pdf (Accessed on: 10 June 2018).

[95] M. Moll, I. Şucan and L. Kavraki. 'Benchmarking Motion Planning Algorithms: An Extensible Infrastructure for Analysis and Visualization'. In: *IEEE Robotics & Automation Magazine* 22.3 (2015), pp. 96–102.

[96] M. Moon. *Amazon crowns winner of first warehouse robot challenge*. 2015. URL: https://www.engadget.com/2015/06/01/amazon-picking-challenge-winner/ (Accessed on: 10 June 2018).

[97] S. Moring. 'Kinodynamic steering using Supervised Learning in RRT'. Master of Science Thesis. Delft University of Technology, 2018.

[98] R. M. Murray, Z. Li and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[99] D. S. Naidu. *Optimal Control Systems*. CRC Press, 2002.

[100] L. Palmieri and K. O. Arras. 'Distance metric learning for RRT-based motion planning with constant-time inference'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 637 –643.

[101] D. Paramkusam. 'Comparison of Optimal Control Techniques for Learning-based RRT'. Master of Science Thesis. Delft University of Technology, 2018.

[102] R. Parasuraman, T. B. Sheridan and C. D. Wickens. 'A model for types and levels of human interaction with automation'. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 30.3 (2000), pp. 286–297.

[103] C. Park, J. Pan and D. Manocha. 'Poisson-RRT'. In: *IEEE International Confernce on Robotics and Automation (ICRA)*. 2014, pp. 4667 –4673.

[104] A. ten Pas and R. Platt. 'Using Geometry to Detect Grasp Poses in 3D Point Clouds'. In: *Proceedings of the International Symposium on Robotics Research (ISRR)*. 2015.

[105] A. Perez et al. 'LQR-RRT*: Optimal Sampling-Based Motion Planning with Automatically Derived Extension Heuristics'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2012, pp. 2537–2542.

[106] F. Pfeiffer and R. Johanni. 'A concept for manipulator trajectory planning'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 1986, pp. 1399 – 1405.

[107] Q-C. Pham, S. Caron and Y. Nakamura. 'Kinodynamic Planning in the Configuration Space via Velocity Interval Propagation'. In: *Robotics: Science and Systems*. 2013.

[108] Q-C. Pham et al. 'Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots'. In: *International Journal of Robotics Research* 36.1 (2016), pp. 44–67.

[109] M. C. Plooij, H. Vallery and M. Wisse. 'Reducing the energy consumption of robots using the Bi-directional Clutched Parallel Elastic Actuator'. In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1512–1523.

[110] L. S. Pontryagin. *Mathematical theory of optimal processes*. CRC Press, 1987.

[111] M. Posa, C. Cantu and R. Tedrake. 'A direct method for trajectory optimization of rigid bodies through contact'. In: *The International Journal of Robotics Research* 33.1 (2014), pp. 69–81.

[112] M. Prats et al. *MoveIt! Workspace Analysis Tools*. URL: http://moveit.ros.org/assets/pdfs/2013/icra2013tutorial/ICRATutorial-Workspace.pdf (Accessed on: 10 June 2018).

[113] H. Prautzsch, Wolfgang Boehm and Marco Paluszny. *Bézier and B-Spline Techniques*. Springer, 2002.

[114] J. Meijer Q. Lei and M. Wisse. 'A survey of unknown object grasping and our fast grasping algorithm C-shape grasping'. In: *Proceedings of the IEEE International Conference on Control, Automation and Robotics (ICCAR)*. 2017, pp. 150–157.

[115] M. Quigley et al. 'ROS: An open-source Robot Operating System'. In: *ICRA Workshop on Open Source Software* (2009).

[116] O. E. Ramos et al. 'Dynamic Whole Body Motion Generation for the Dance of a Humanoid Robot'. In: *IEEE Robotics and Automation Magazine* 22.4 (2015), pp. 16–26.

[117] A. V Rao. 'A survey of numerical methods for optimal control'. In: *Advances in the Astronautical Sciences* 135.1 (2009), pp. 497–528.

[118] N. Ratliff et al. 'CHOMP: Gradient optimization techniques for efficient motion planning'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2009, pp. 489–494.

[119] J. H. Reif. 'Complexity of the mover's problem and generalizations'. In: *IEEE Symposium on Foundations of Computer Science*. 1979, pp. 421 –427.

[120] S. Ren et al. 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149.

[121] N. Rikovitch and I. Sharf. 'Kinodynamic Motion Planning for UAVs: A Minimum Energy Approach'. In: *AIAA Guidance, Navigation and Control (GNC) Conference* (2013).

[122] H. Robbins. 'Some aspects of the sequential design of experiments'. In: *Bulletin of the American Mathematical Society* 58 (1952), pp. 527–535.

[123] O. Russakovsky et al. 'ImageNet Large Scale Visual Recognition Challenge'. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252.

[124] J. Sacks et al. 'Design and Analysis of Computer Experiments'. In: *Statistical Science* 4 (1989), pp. 409–423.

[125] C. Samson, M. Le Borgne and B. Espiau. *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom, 1991.

[126] E. Schmerling, L Janson and M. Pavone. 'Optimal sampling-based motion planning under differential constraints: the drift case with linear affine dynamics'. In: *IEEE Conference on Decision and Control*. 2015, pp. 2574 –2581.

[127] J. Seipp et al. 'Automatic Configuration of Sequential Planning Portfolios'. In: *AAAI* (2015), pp. 3364–3370.

[128] L. Sentis and O. Khatib. 'Synthesis of whole-body behaviors through hierarchical control of behavioral primitives'. In: *International Journal of Humanoid Robotics* 2.4 (2005), pp. 505–518.

[129] K. G. Shin and N. D. McKay. 'Minimum-time control of robotic manipulators with geometric path constraints'. In: *IEEE Transactions on Robotics* 30.6 (1985), pp. 531 –541.

[130] A. Shkolink, M. Walter and R. Tedrake. 'Reachability-guided sampling for planning under differential constraints'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2009, pp. 2859–2865.

[131] T. Siméon, J-P. Laumond and C. Nissoux. 'Visibility based probabilistic roadmaps for motion planning'. In: *Advanced Robotics* 14.6 (2000), pp. 477–493.

[132] K. Simonyan and A. Zisserman. 'Very deep convolutional networks for large-scale image recognition'. In: *arXiv preprint arXiv:1409.1556* (2014).

[133] I. A. Şucan and S. Chitta. *MoveIt! Motion Planning Framework*. URL: http://moveit.ros.org (Accessed on: 10 June 2018).

[134] I. A. Şucan and L. E. Kavraki. 'Kinodynamic motion planning by interior-exterior cell exploration'. In: *Algorithmic Foundation of Robotics* VIII (2009), pp. 449–464.

[135] I. A. Şucan, M. Moll and L. E. Kavraki. 'The Open Motion Planning Library'. In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.

[136] A. Tallavajhula et al. 'List prediction applied to motion planning'. In: *IEEE International Conference On Robotics and Automation (ICRA)*. 2016, pp. 213 –220.

[137] Y. Tassa, T. Erez and E. Todorov. 'Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization'. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 4906–4913.

[138] R. Tedrake et al. 'LQR-trees: Feedback motion planning via sums-of-squares verification'. In: *International Journal of Robotics Research* 29.8 (2010), pp. 1038–1052.

[139] Tsianos, K. I. and Şucan, I. A. and L. E. Kavraki. 'Sampling-Based Robot Motion Planning: Towards Realistic Applications'. In: *Computer Science Review* 1.1 (2007), pp. 2–11.

[140] J. R. R Uijlings et al. 'Selective search for object recognition'. In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[141] C. Urmson and R. Simmons. 'Approaches for Heuristically Biasing RRT Growth'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2003, pp. 1178 –1183.

[142] S. Vijayakumar and S. Schaal. 'Incremental Online Learning in High Dimensions'. In: *Neural Computation* 17 (2005), pp. 2602–2634.

[143] D. J. Webb and J. van den Berg. 'Kinodynamic RRT*: Asymptotically Optimal Motion Planning for Robots with Linear Dynamics'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 5054 –5061.

[144] F. Weisshardt and F. Koehler. 'Automatic Testing Framework for Benchmarking Applications'. In: *Proceedings of ISR 2016: 47st International Symposium on Robotics*. 2016, pp. 1–6.

[145] M. Wisse. *Factory in a Day Project (FiaD)*. 2013. URL: http://www.factory-in-a-day.eu/ (Accessed on: 10 June 2018).

[146] W. J. Wolfslag et al. 'Learning robustly stable open-loop motions for robotic manipulation'. In: *Robotics and Autonomous Systems* 66 (2015).

[147] A. Yershova and S. M. LaValle. 'Improving Motion Planning Algorithms by Efficient Nearest-Neighbor Searching'. In: *IEEE Transactions on Robotics* 23 (2007), pp. 151–157.

[148] M. Zucker. *Approximating State-Space Obstacles for Non-Holonomic Motion Planning*. Tech. rep. Robotics Institute, Carnegie Mellon University, 2006.

A

Motion Module for the Amazon Robotics Challenge 2016 - Team Delft

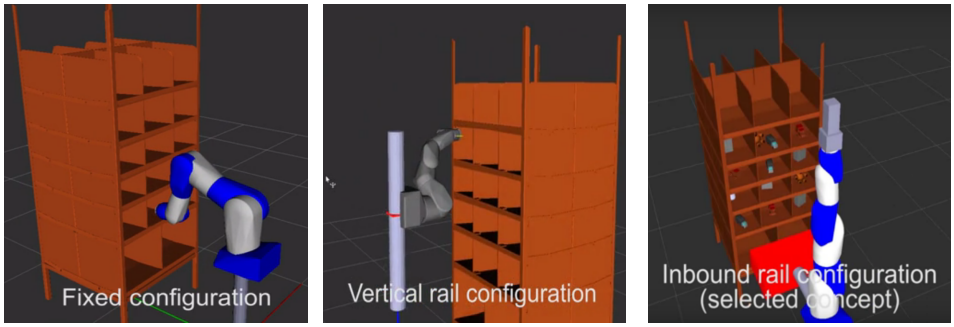
The Team Delft motion module for Amazon Robotics Challenge (ARC) 2016 was designed to ensure our robot could move to all commanded locations to accomplish both the picking and stowing tasks in the challenge. In this appendix, the design and implementation details of the motion module are further elaborated as supporting material to the contents in Chapter 3. Some of the key observations from the design process and some of the limitations of the (software) design assumptions are also presented. It is assumed in this text that the reader has a basic understanding of the Robot Operating System (ROS) software framework[1] and the MoveIt! motion planning tool [133].

As highlighted in Section 3.3.5 in Chapter 3, the robotic system was designed by a systematic consideration of the different requirements of the challenge. From a motion planning perspective, the key requirement was to guarantee maximum reachability within the operational workspace with adequate maneuverability within the bins for object manipulation. In the following section, we explain the procedure that was followed to arrive at the eventual robotic system configuration.

A.1. Robotic System Selection

We considered multiple robotic system configurations as indicated in Figure A.1 keeping in mind the rules of the challenge for building the *robot cell*. We subsequently evaluated each system's workspace reachability using the MoveIt! workspace analysis tools [112].

[1] https://wiki.ros.org/



(a) Robot mounted on a fixed pedestal. (b) Robot mounted on an upward translating rail. (c) Robot mounted on an inward translating rail.

Figure A.1: A few robot system configurations considered for evaluation before deciding on the robot hardware.

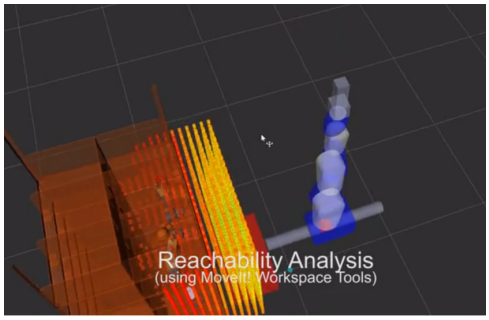


Figure A.2: Reachability Analysis results of the selected robotic system configuration. The color of the marker in the figure indicates the number of IK solutions available to reach the corresponding cartesian location in the robot cell. Red markers are for locations that have very few IK solutions.

It was observed that the configuration shown in Figure A.1c would ensure that we can reach all bins with adequate maneuverability. The reachability analysis results on this robot system configuration is presented in Figure A.2.

An important aspect that can be observed in the workspace analysis is that some of the deep corners of the bins have no markers indicating those regions are not accessible with this system configuration. This limitation was compensated by slightly increasing the length of the suction tool (represented by the fictitious solid gray block in Figure A.1c). A further increase in the size of the reachable region was achieved with the help of a pneumatic mechanism to extend the suction cup. Although, the main purpose of this mechanism was to achieve different orientations of the suction cup to ensure a stable grasp.

A.2. MOTION MODULE DESIGN

The ARC 2016 consisted of two main tasks. The picking task required the grasping of a target object in the presence of other objects in the bin and placing the grasped object into a tote. The stowing task comprised of moving a target object in the tote to one of the bins in the shelf. To fulfill these tasks, the motion module was built on two fundamental motion primitives namely, coarse motions and fine motions.

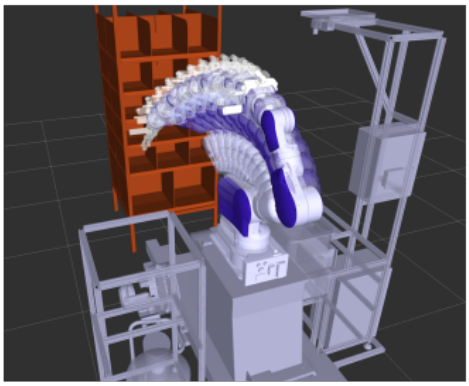


Figure A.3: Coarse motion as a trail from “Bin D” to “Home”.

Coarse motions were essentially offline generated trajectories between pre-defined start and goal positions in the operational workspace of our system. On the other hand, fine motions involved online (cartesian) path planning for performing object manipulation in the bins or the tote. In the following sections, these two primitives will be explained in further detail.

A.3. COARSE MOTIONS

The functional robot workspace for the ARC was static. This formed the basis for the coarse motion primitive. In other words, there were no dynamic obstacles that would obstruct the path of the robot after the robot had started moving. Thus, we defined a coarse motion primitive as a trajectory that can be computed offline between a predefined start and goal location.

Subsequently, we implemented a *trajectory cache*[2] which is populated with trajectories between around 250 different start and goal configurations for the robot. We call these configurations as Master Pose Descriptors. In principle, these are robot joint states set at appropriate values in front of each bin of the shelf. Our choice of having the camera mounted on the manipulation tool entailed that we have two master pose descriptors per bin, namely, the camera master pose descriptor and bin master pose descriptor. Similar master pose descriptors were also defined for the tote drop-off locations. All trajectories that were used during APC 2016 were generated using RRT-Connect randomized path planner via MoveIt! The other planner option we tried was RRT-star which did not have any significant benefit over RRT-Connect in the given planning environment. An example of a coarse motion trail from one of the bins to the *home* position of our setup is shown in Figure A.3.

[2]The use of cache is a misnomer because in the current implementation, we do not compute a coarse motion online, if a requested coarse motion does not exist in the cache. A trajectory database would have been a better name, in hindsight.

A.4. FINE MOTIONS

Fine motions were the only part of the APC motion module that involved online (cartesian) path planning. The idea of fine motions is a simplified implementation of the approach in the standard pick and place pipeline of MoveIt!, where cartesian path planning is used during the pre-grasp approach and the post-grasp retreat stage of the pipeline. The simplification is the fact that we remove the evaluation of the reachable and valid pose filter stage of the standard pick and place pipeline.

This process of finding and evaluating reachable and valid poses are done in a two-step filtering. The first filtering step is done in the grasp synthesizer module where impossible grasps are eliminated heuristically. The second filtering step consists of multiple MoveGroup API calls to `computeCartesianPath` after some key cartesian waypoints are evaluated for collisions using the `getPositionIK` service. Further details on this step will be more coherent to read once the background information concerning our grasp strategy is explained. It is important to highlight that, there is a serious limitation with our approach when we consider object manipulation inside a bin in general. This is because, we disallow any kind of collision with neighboring objects inside a bin (or tote) and also end up in situations where no valid grasp candidates are found. However, allowing for useful collisions with other objects in a bin is certainly something we would consider in the future.

A.4.1. GRASP STRATEGY

From a motion perspective, the grasp strategy for all objects in APC 2016 consisted of a combination of linear segments. We call these segments as *Approach*, *Contact*, *Lift* and *Retreat*. The segment names are indicative of the corresponding motions that those segments are meant for. Once the cartesian pose of the object of interest (for both pick and stow tasks), in the object reference frame is estimated by the pose estimation algorithm, the grasp synthesizer uses this pose to generate a set of key waypoints for the start and end of each segment and in some cases more than just a pair of waypoints to limit any possibility of configuration changes while manipulating the object in the bin. These cartesian waypoints form an important input to the second step of grasp pose filtering and the fine motion generation.

A.4.2. MOTION SEGMENT GENERATION

The key waypoints corresponding to Approach, Contact, Lift and Retreat along with a potential grasp candidate are all input to the motion generation module where the following checks are conducted to generate the complete sequence of motions:

1. The grasp candidate, and the key waypoints are sequentially checked for collision using the `getPositionIK` service call. If any one of them is in collision, the corresponding grasp pose and the key waypoints are all discarded due to collision(s).

2. Once all the key waypoints are collision free, each linear motion segment is computed using the MoveGroup API, `computeCartesianPath` with collision checking enabled. Similar action is taken if any of these segments are in collision.

3. If all the linear motion segments are collision free, a final planning call is done using the MoveGroup API, `plan`[3]. This is required because, the final joint configuration at the end of the Retreat segment resulting from the call to `computeCartesianPath` need not necessarily match the starting joint configuration in front of the bin or tote from where the coarse motions start. This is natural because of the redundancy in the system. The call to `plan` is precisely to ensure that such a mismatch does not exist which would otherwise lead to a motion safety violation [4].

These steps ensure that all the desired motion segments for manipulating the object of interest are generated as required and are collision free. These segments are now stitched together before being executed on the robot. The stitching module is explained in the following section.

A.5. MOTION STITCHING AND EXECUTION

The motion stitching module accepts all the motion segments that are generated as explained in the previous section. Additionally, the coarse motion trajectories from the corresponding bin to the tote drop-off location (or vice-versa for stowing) are also input to the stitching module. The stitching process, in principle, is the process of combining the joint state configurations from each segment into one single motion plan and time parameterizing it so that it results in a executable trajectory for the robot. We use the `computeTimeStamps` from the TrajectoryProcessing API for this purpose. This also allowed us to use object specific velocity scaling (for instance, moving at low speeds when carrying heavy objects such as the dumbbell, socks and paper towels). An additional advantage of stitching multiple motion segments was that we could get rid of overheads such as goal tolerance checks at the end of each segment execution. This indeed provided us quite a significant time gain while executing the motions. Finally, the time parameterized trajectories were executed using the MoveGroup API, `execute`.

The final and a critical component of our motion module is the I/O handling to ensure the end effector (suction or pinch) is actuated at the right times along the trajectory.

A.5.1. INPUT-OUTPUT (I/O) HANDLING

In the ARC setting, it was critical to ensure the I/O was activated accurately and in a timely manner. For instance, the vacuum pump needed a couple of seconds before full suction power was realized. However, turning the suction on too early could pose significant problems while approaching certain objects which have a loose plastic covering. Basically, such objects would get suctioned in way too early before the approach was completed leading to either an unstable or a failed grasp. In order to address this, we built a custom trajectory tracking module based on the `/joint_states` topic which provided continuous joint state information.

The trajectory tracking module used a gradient descent based approach to detect events along the trajectory based on distance to key joint state waypoints. These events and waypoints have a direct association to the key waypoints of the motion segments that we

[3] RRT-Connect is used here as the planner configuration due to fast solution times.

[4] A motion safety violation is triggered whenever the starting configuration of the robot does not match the starting configuration in the trajectory that is about to be executed.

described earlier. As a matter of fact, these events are created exactly at the same point in the code where the key waypoints are created for the motion segments. The term “event” is used to emphasize that they are actual events along the trajectory such as beginning of approach segment, beginning of contact segment, end of retreat segment and so on. All these events also are used as feedback to evaluate the success or failure of a grasp. For instance, the vacuum sensor is read at the end of the retreat from the bin (or tote) to determine whether the grasp succeeded.

A commonly used alternative for I/O handling in MoveIt! is the definition of I/O as joints with minimal displacement and providing them target joint values at appropriate times via regular planning calls. We do not use this approach as it does not guarantee adequate synchronization with the events that we define along the trajectory. Another note of caution is that the implementation of the trajectory tracking module was not completely straightforward due to the choice of using a *service* based sequential software architecture instead of the asynchronous *action* based architecture for the motion module. The main reason for using a sequential architecture was to avoid any potential race conditions that could occur in asynchronous designs and thus impact robustness.

NOTES

- The open source software repository hosting the code for the motion module and the rest of the ARC 2016 software is available at:
https://github.com/warehouse-picking-automation-challenges/team_delft
- The documentation for the various APIs referred in this text can be accessed via:
http://moveit.ros.org/code-api/

B

RRT-CoLearn: Scalability considerations

In Chapter 6, we presented the RRT-CoLearn algorithm to approximate both the optimal cost and a control parameterization to connect state-pairs. The effectiveness of the approach was shown on the 2-dimensional state space of the simple pendulum system. In this appendix, we present some preliminary directions that have been explored in order to extend the applicability of RRT-CoLearn to planning in higher dimensional state spaces. Three important perspectives are presented here: (i) formulation of the system dynamics for indirect optimal control, (ii) state space coverage and (iii) better machine learning. For this purpose, we consider a planar manipulator with two degrees of freedom (see Figure B.1) and consequently, a 4-dimensional state space and an 8-dimensional learning space.

B.1. System dynamics for indirect optimal control

In this section, we will discuss the relation between the formulation of system dynamics and the resulting parameterization of the control input. The equations of motion for the

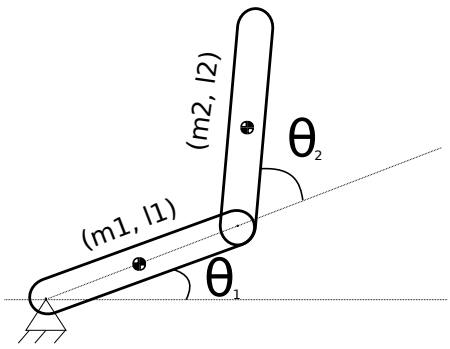


Figure B.1: Schematic of planar a two link manipulator.

simple pendulum system used in Chapter 6 were derived with the well known Lagrangian formulation [36]. The same principle will be used here to derive the equations of motions of the two link manipulator using the angles and angular velocities as the generalized coordinates.

The first step is to formulate the Lagrangian $\mathscr{L}$ which is the difference between the kinetic and potential energies of a system:

$$\mathscr{L} = T(\mathbf{q}, \dot{\mathbf{q}}) - V(\mathbf{q}) = \frac{1}{2}\dot{\mathbf{q}}M(\mathbf{q})\dot{\mathbf{q}} - V(\mathbf{q}) \tag{B.1}$$

where T is the kinetic energy, V is the potential energy, $\mathbf{q}$ and $\dot{\mathbf{q}}$ indicate the generalized positions and velocity vectors respectively. Subsequently, the equations of motions are found using:

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{q}_i} - \frac{\partial \mathscr{L}}{\partial q_i} = 0 \tag{B.2}$$

Following the steps in [98] and assuming motor torques τ_1 and τ_2 being applied at each joint, the equations of motion for the two link manipulator are found as:

$$\begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \ddot{\theta_1} \\ \ddot{\theta_2} \end{bmatrix} = \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ M^{-1}(\mathbf{q})\mathbf{u} - M^{-1}(\mathbf{q})B(\mathbf{q}, \dot{\mathbf{q}})[\dot{\theta_1}\,\dot{\theta_2}]^T) \end{bmatrix} \tag{B.3}$$

where, $M(\mathbf{q})$ and $B(\mathbf{q}, \dot{\mathbf{q}})$ are matrices containing information about the physical properties of the manipulator and $\mathbf{u} = [\tau_1\,\tau_2]^T$ represents the control input vector. Let $x = [\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2}]$ be the state vector and the cost function J defined as in Chapter 6:

$$J(x(t), u(t)) = \int_0^{t_f} C(x(t), u(t))\mathrm{d}t \tag{B.4}$$

Let $\lambda = [\lambda_{\mathbf{q}}; \lambda_{\dot{\mathbf{q}}}]$ be the costate vector with $\lambda_{\mathbf{q}} = [\lambda_{\theta_1}\,\lambda_{\theta_2}]^T$ and $\lambda_{\dot{\mathbf{q}}} = [\lambda_{\dot{\theta_1}}\,\lambda_{\dot{\theta_2}}]^T$. The optimal control Hamiltonian can thus be defined as:

$$\mathscr{H}(x, \lambda, \mathbf{u}) = C + {\lambda_{\mathbf{q}}}^T\dot{\mathbf{q}} + {\lambda_{\dot{\mathbf{q}}}}^T\ddot{\mathbf{q}} \tag{B.5}$$

Substituting Eq. B.3 in Eq. B.5, we get:

$$\mathscr{H} = C + {\lambda_{\mathbf{q}}}^T\dot{\mathbf{q}} + {\lambda_{\dot{\mathbf{q}}}}^T\left(M^{-1}(\mathbf{q})\mathbf{u} - M^{-1}(\mathbf{q})B(\mathbf{q}, \dot{\mathbf{q}})[\dot{\theta_1}\,\dot{\theta_2}]^T)\right) \tag{B.6}$$

Finally, with $C = 1 + \frac{1}{2}\mathbf{u}^T\mathbf{u}$ the optimal input $\mathbf{u}^*$ can be solved for as follows:

$$\frac{\partial \mathscr{H}}{\partial \mathbf{u}} = \mathbf{u}^* + {\lambda_{\dot{\mathbf{q}}}}^T\left(M^{-1}(\mathbf{q})\mathbf{u} - M^{-1}(\mathbf{q})B(\mathbf{q}, \dot{\mathbf{q}})[\dot{\theta_1}\,\dot{\theta_2}]^T)\right) = 0 \tag{B.7}$$

$$\mathbf{u}^* = -{\lambda_{\dot{\mathbf{q}}}}^T\left(M^{-1}(\mathbf{q})\right) \tag{B.8}$$

Following similar steps as detailed in Chapter 6, the data generation, learning and the planning steps of RRT-CoLearn can be completed. A key point to note here is in the equation for the optimal control input $\mathbf{u}^*$ in Eq. B.8. As opposed to the case with the single pendulum,

the expression for $\mathbf{u}^*$ is not only a function of the co-states but also has a dependency on the configuration of the manipulator. This state dependency is to be expected due to the dynamical coupling between the links. However, since we only learn a mapping between a state pair and the corresponding co-states, this state dependency could hinder the ability of the learned model to generalize and consequently lead to poor performance in the steering phase of RRT-CoLearn. Preliminary studies in this direction are available in [101]. Further research is necessary to clearly understand how this dependency would impact learning.

Interestingly, this dependency does not appear if the Hamiltonian[1] formalism is used instead of the Lagrangian formalism to derive the equations of motion. In the Hamiltonian formalism, generalized momenta are used in place of generalized velocities to represent the evolution of system dynamics. The space of positions and momenta is some times also referred to as phase space. Further, the system Hamiltonian can be derived from the system Lagrangian in Eq. B.1 using the Legendre transformation [6] as follows:

$$\mathscr{H}_{\text{system}} = \sum_i \dot{q}_i \frac{\partial \mathscr{L}}{\partial \dot{q}_i} - \mathscr{L} \tag{B.9}$$

Let $\mathbf{p}$ denote the generalized momenta. According to [6], the relation between the generalized momenta and the generalized velocities is given by $p = \frac{\partial \mathscr{L}}{\partial \dot{q}_i} \implies \dot{\mathbf{q}} = M^{-1}(\mathbf{q})\mathbf{p}$. With this relation, the evolution of the generalized positions and momenta are given by the following equations [6]:

$$\dot{\mathbf{q}} = \frac{\partial \mathscr{H}_{\text{system}}}{\partial \mathbf{p}} \tag{B.10}$$

$$\dot{\mathbf{p}} = -\frac{\partial \mathscr{H}_{\text{system}}}{\partial \mathbf{q}} + \mathbf{u} \tag{B.11}$$

Let the costates corresponding to the generalized momenta be represented by $\lambda_{\mathbf{p}} = [\lambda_{p_1}\, \lambda_{p_2}]^T$. Together with $\lambda_{\mathbf{p}}$ from above, the control Hamiltonian can be defined as:

$$\mathscr{H} = C + {\lambda_{\mathbf{q}}}^T \dot{\mathbf{q}} - {\lambda_{\mathbf{p}}}^T \frac{\partial \mathscr{H}_{\text{system}}}{\partial \mathbf{q}} + {\lambda_{\mathbf{p}}}^T \mathbf{u} \tag{B.12}$$

Following the same procedure as earlier the optimal control input $\mathbf{u}^*$ is computed as:

$$\frac{\partial \mathscr{H}}{\partial \mathbf{u}} = \frac{\partial C}{\partial \mathbf{u}} + {\lambda_{\mathbf{p}}}^T = 0 \tag{B.13}$$

From the above equation, it is clear that $\mathbf{u}^* = -\lambda_{\mathbf{p}}$. In other words, the optimal control input is parameterized by the costates corresponding to the generalized momenta. This is certainly desirable compared to Eq. B.8 as the parameterization is state independent. The benefits of the Hamiltonian formalism and its effects on the learning performance could be an interesting avenue to investigate further in the context of scaling up RRT-CoLearn to higher dimensional state spaces.

[1] This Hamiltonian is different from the control Hamiltonian. We will refer to this as the system Hamiltonian to avoid confusions.

B.2. State space coverage

In this section, we briefly discuss another aspect that could influence the coverage of the state space during data generation when we apply RRT-CoLearn to higher dimensional state spaces. In Chapter 6, it was highlighted that a crucial constraint arises from the nature of the optimal control problem in RRT-CoLearn, i.e.:

$$\mathscr{H}^*\left(x(0), \lambda(0)\right) = 0 \tag{B.14}$$

In the two dimensional state space of the pendulum, we could exclusively solve for the roots of this equation and hence ensure the complete range of the costates is covered. This is important because, the initial costates characterize the (locally) optimal trajectories that we learn from. However, when we consider the 4-dimensional state space of the two link manipulator, the solution to this constraint equation is non-trivial.

Basically, the optimal $\mathscr{H}^*$ is a function of the state and the costate variables. In case of the 4-dimensional state space of the two link manipulator, we are left with 4 free costate variables to choose from to ensure the constraint is satisfied. This can be addressed by uniformly randomly sampling three of the four costate variables and solving for the last one using a numerical solver. However, there is no implicit guarantee on the uniformity of the distribution of this costate when a numerical solution is used. This could in turn impact the coverage of the state space and hence localize the learning to only those regions that happened to be created from the numerical solutions. Coming up with implicit ways to address this problem (via factorization of $\mathscr{H}^*$ for instance) could be a potential direction to pursue further.

Another relevant aspect related to state space coverage arises from the short-time reachability assumption for RRT-CoLearn.Therefore, more effort is needed to establish the applicability of RRT-CoLearn for underactuated systems. For a preliminary study in this direction, see [97].

B.3. Machine learning improvements

A simple learning algorithm (k-Nearest Neighbor (kNN)) was used for learning the mapping between cost and costates for the RRT-CoLearn on the simple pendulum system. While, this was sufficient to establish the proof of concept, this only worked *after* the data was explicitly processed to remove the bias caused by *similar* samples. This process of data bias removal involved the tuning of a non-intuitive heuristic and the process itself was also quite time consuming. Additionally, another heuristic (the reachability heuristic) was also used to ensure we avoid erroneous predictions due to interpolation or extrapolation.

For the problem of bias removal, recent machine learning techniques such as deep generative models could be considered. The benefit of such methods is that instead of removing the bias, the underlying distribution in the data is learned (given a certain prior). In a crude sense, the many-to-one mapping between costates and a given state pair can also be learned. This could potentially be useful when considering functionalities such as (online) collision avoidance when a certain predicted low-cost steering could get invalidated due to collisions. A primary work in this direction has been pursued in [93].

C

CHAPTERS WITH SHARED AUTHORSHIPS

The work done and the results presented in Chapter 2, Chapter 3 and Chapter 6 are outcomes of close collaborations with different authors. Here, the contributions to each of these chapters from the author of this dissertation is mentioned briefly.

CHAPTER 2: TUNING OF PATH PLANNERS

The idea of this contribution was a result of a discussion between Ruben Burger, Mukunda Bharatheesha and Robert Babuška during the time when Ruben Burger was conducting his MSc thesis research. Ruben Burger implemented the framework of connecting the SMAC tools to the library of different planners in MoveIt! Mukunda Bharatheesha contributed to different parts of this chapter such as relevant literature, problem formulation, the relevant inputs to the SMAC tools for the UR5 difficult pick and place problem and discussion and tabulation of the results.

CHAPTER 3: AMAZON PICKING CHALLENGE 2016

The authors of this chapter were all a part of *Team Delft* that won the Amazon Picking Challenge 2016. Mukunda Bharatheesha lead the motion planning team and contributed to the development of multiple software modules (such as `motion_executor`, `manipulation_planner`, `trajectory_cache`) that were a part of the entire motion module of the robotic system. The motion planning module is further elaborated in Appendix A.

CHAPTER 6: LEARNING INDIRECT OPTIMAL CONTROL

The main idea of for this chapter emerged as an outcome of a discussion between Wouter Wolfslag and Mukunda Bharatheesha. While exploring avenues of extending the ideas of Chapter 5 to also approximate the steering inputs for a state space RRT, Wouter Wolf-

slag suggested the possibility of using Indirect Optimal Control to generate the data for RRTLearn. Wouter Wolfslag worked on the data generation and the initial completeness considerations for RRT-CoLearn and Thomas Moerland contributed to the dataset cleaning section of the chapter in collaboration with Wouter Wolfslag. Mukunda Bharatheesha contributed to the RRT part of the chapter together with conducting the different experiments. Mukunda Bharatheesha also contributed towards the probabilistic completeness aspects that are presented in this chapter.

ACKNOWLEDGEMENTS

It is a humbling feeling as I take the opportunity to thank several people that have enabled and supported me to reach this stage of my academic life. All of you adorn the various facets of a PhD process that make this journey a truly fulfilling experience.

First and foremost, I am grateful to my promotor Prof. Dr. Ir. Martijn Wisse for providing me with the opportunity to pursue this research. Martijn, you believed in me and have been a constant source of encouragement. The exciting and engaging discussions we had enabled me to learn and explore various fundamental topics in motion planning. The constructive criticism and feedback you gave about my work always inspired me to do better. Your outright practicality in suggesting solutions to some problems and honesty were sometimes bewildering to me. I learned some life lessons nonetheless. Also, you provided me with an opportunity to be a part of two fantastic projects in Factory in a Day (FiaD) and Amazon Robotics Challenge (ARC). Contributing to these projects was a lot of fun and resulted in some very good memories with amazing people. I will cherish them for years to come! Also, I am immensely thankful to you for the faith you showed in me while I was going through some tough personal times. Your words of advice and supportive actions have played a major role in me successfully getting to this point in life, professionally and personally.

Next, I express my thanks to all the revered members of my PhD thesis committee who took the time and effort to review my work and provide feedback for improvement.

I also extend my thanks to Prof. Dr. Robert Babuška and Dr. Wouter Caarls for the interesting discussions and remarks in the early stages of this research. Particularly, the discussions during the *RAP meetings* helped me refine my research focus and gain insights into topics such as machine learning and optimal control methods. Robert, despite your busy schedules and interactions with so many students, you were regularly in touch with my work. Thank you for taking time to point out some relevant papers you would come across related to motion planning research. Wouter, thank you for always being available to help with all sorts of beginner level questions on Linux, ROS and CMake. Prof. Dr. Pieter Jonker, thanks for your inputs and discussions during the first year of my PhD.

The results achieved in this thesis would not have been possible without collaboration with two wonderful colleagues, Ir. Wouter Wolfslag and Ir. Gijs van der Hoorn. I have learned and continue to learn a lot from you guys. Wouter, we have had many enjoyable discussions during tea, lunch and *foosball* meetings. Optimal control, machine learning, *beautiful* math tricks, meticulously planning work out exercises at the gym with an associated diet, *nerd* humor, sports, life, universe and the list of topics we spoke about is practically endless! Thank you for not only being a colleague but also a good friend to me all these years.

Gijs, every time I have talked to you, I have been enriched with some new software tools and knowledge (particularly related to ROS). It has always astonished me how you quickly recall and point to the most relevant tool from your oceans of software knowledge! Thank you for the time and patience in explaining so many different software development concepts

on *how things are supposed to be done* and politely pointing out my non-compliance to it. Also, thank you very much for bailing me out when I messed up my Linux installations at critical times! You have been a great example of how to selflessly contribute to the growth and success of various individuals and projects. I aspire to emulate these qualities in my life as well.

My thanks also to Ruben Burger, Carlos Hernández Corbato and Thomas Moerland for their valuable help in realization of the results presented in some of the chapters in this thesis. I also thank Maryam Sharify for her support and help with making the cover design for this thesis.

I thank Dr. Florent Lamiraux, LAAS-CNRS, Toulouse for hosting me at the *Gepetto* group for six months. Florent, the opportunity to work with the HPP path planner and its state space extensions helped me understand the intricacies of a generic path planning framework development. Along with this, the feedback I received from you, Dr. Nicolas Mansard, Dr. Michel Taïx and Dr. Jean-Paul Laumond about improvements on my research was highly valuable. Although brief, the insightful discussions with Dr. Quang-Cuong Pham about alternative methods for kinodynamic planning broadened my view of the research field. My subsequent interaction with Nirmal Giftsun was equally insightful. Nirmal, your patience and persistence served as motivating factors during the long evenings solving various challenges we faced in realizing robot demonstrations. Thank you buddy. The first few weeks in Toulouse progressed smoothly thanks to Dr. Joseph Mirabel. Joseph, thank you for helping me figure out basic necessities to settle down in Toulouse. I have met a few *vim fanatics* but I haven't yet met someone who is as *plugged in to vim* as you are. My stay in Toulouse was also made enjoyable by a big group of friends and colleagues - Andreas, Andrea, Christian, Nemanja, Daniel, Brigita, Renaud, Harmish, Ganesh, Mathieu, Maximilian, Olivier, Mehdi, Ixchel, Naoko, Justin, Myléne, Anguéron and Típhain. *Merci á tous!*.

Reflecting back on the beginning of this journey at the Delft Biorobotics Laboratory (DBL), I am deeply thankful to the unwavering trust and belief shown by Maja Rudinac. Maja, without your support and encouragement, I don't think I would have even started this journey. Starting off as a research assistant, I enjoyed every moment of working for the RoboCup@Home team with Aswin, Machiel, Susana, Floris and Guus. Thanks to all of you for the great memories!

Contributing to different projects during the course of my PhD also lead to a great camaraderie with several people associated with these projects. At first, I would like to thank the FiaD project partners that I actively collaborated with. Katharina, Florian, Dr. Emmanuel Dean and Prof. Dr. Gordon Cheng (TUM), your support in the hardware and software deployment of the Cellular Skin on the robot arm used in this thesis was invaluable. You were always available for help related to both the skin hardware and software, be it in person or remotely. Then, I would like to thank Luca (PAL Robotics) and Robert (UR) for their helpful pointers on the real-time aspects of using the UR robot with ROS. I also thank Mirko Bordignon and Felix Meßmer (Fraunhofer IPA) and Willem Verleysen (Materialise) for integration support at different phases in the project. Most importantly, thank you Wibke and Dunja for taking care of the organizational and management aspects of such a large project.

Jan and Bas, your expertise in hardware design, prototyping and production was instrumental in the timely realization of various interim and final goals of the Amazon Robotics Challenge. Your support was so reliable that I could practically forget about hardware and just focus on the motion planning software interfaces. And once the hardware was ready, it was *plug and play*! Thanks for all the fun times during the vacuum suction tests, tolerating all the bloopers that lead to hardware damage and also great company. A shout out to all the fantastic members of the ARC team - Ruben, Carlos, Jethro, Wilson, Maarten, Jeff, Hans, Mihai, Xander, Jihong, Kanter and Ronald, you guys were awesome to work with. The entire journey from the start to the end of the challenge is a truly vivid memory!

Complementing the research activities during this thesis were also the different opportunities I got to interact with students at TU Delft. Martijn, thank you once again for the teaching opportunity in the Humanoid Robots Course. It was an enjoyable and fun experience to interact with a large group of motivated MSc students. Besides, I also enjoyed supervising multiple MSc theses in the past years - Ruben, Stefan, Deepak, Hugo and Wouter, it was a pleasure to work with all of you.

A special thanks goes out to the administration personnel of our department and the graduate school. Diones, Sabrina, Hanneke, Mirjam and Karin, thank you for your super quick processing of all my (sometimes complicated) travel and administration requests and your ever smiling and enthusiastic presence at the secretariat. Thanks also to Mascha Toppenberg and Erica Radelaar for their untiring efforts in organizing activities and courses for enhancing social interactions between PhD students from different groups.

I have been lucky to experience a friendly atmosphere at DBL, thanks to some fabulous colleagues. Jeff, Joost, Michiel, Wouter, Carlos, Ruben, Martin, Patricia, Saher, Heike, Ivan, Daniel, Andy, Maryam and Christian, thank you all for decorating this journey with your heartening interactions. Friday evening drinks, board games, foosball, kubbs, cricket, bowling, karting, paint ball - I thoroughly enjoyed all these activities and these memories always cheer me up. *Bedankt, lieve mensen!* A special thanks also to former colleagues Berk, Qujiang, Boris, Xin and Jun for the warm conversations.

Life outside the university has also been equally delightful thanks to some wonderful friends. My roommates for the first two and a half years - Subbu, Susana and Anna, thank you for being like a second family. Subbu-san, our sincerity towards preparing authentic Indian cuisine (every evening) and consuming it without any traces of it for the next day still amazes me. Add to that some good humor and philosophical discussions, it was a great way to ease off the stress of the daily grind. Thank you very much for some splendid culinary experiences! Susana, thanks to you too for joining our culinary bandwagon with your Spanish delicacies. Vijay, its nice to have known you for more than 8 years now in Delft. There is always a feeling of calmness when I speak to you and your astronomical levels of patience continues to baffle me! Jeff and Amanda, thank you guys for inviting me for the curry evenings at your house and I am glad I could share some of my cooking experiences with you. Its always a pleasant feeling to recollect these memories. I also would like to thank my friends and acquaintances at Badminton Club DropShot in The Hague. Particularly, Max Umeh, Anwar Orie, Mark Speckens and Martijn van Leuven, you guys made the badminton playing experience so much fun. Also, my Dutch skills improved significantly during my time playing for DropShot. I also thank Aswin for the running evenings and the discussions about effectively finding peace of mind.

I would like to thank teachers and mentors from my previous education in India. Particularly, Mr. M. Gururaja, Mr. C. S. Sampangiram, Mrs. R. K. Karunavathi and Mr. Sudhindra Haldodderi. I have had the opportunity to learn from all of you and you ignited the spark in me to pursue further education after my Bachelor's degree. I am also thankful to my mentors at Robert Bosch India who helped me hone my technical abilities in a professional environment, develop good communication skills and professional ethics.

I express my thanks also to my closest friends who have been an integral part of my social life. NP, PC, Bala and Prabha, thanks to all of you for making this journey of life so beautiful with your evergreen friendships.

A PhD journey is a long process and always involves good and difficult times. Words cannot express how thankful I am to have had people who helped me through some tough times. Paula, Maureen, Martijn, Suhas, Karthik, Michiel, Ineke and Sangeetha, I am indebted to you for everything you did for me during these times. The strength to get this far wouldn't have been there without you!

Getting to Delft and pursuing my Master's and PhD studies wouldn't have been possible without the boundless support, motivation and love of my family. My beloved parents, Smt. Sujatha and Shri. H. R. Bharatheesha, first of all, I want to thank you for all the efforts you put through several years in the past so that this journey was even practically possible. You gave me the beautiful gift of education and helped me grow up as a responsible individual. You gave me the freedom to explore and guided me when I was lost. I am eternally thankful to you for your patience, love and belief in me. My brother Suhas, thank you for all the care and practical advice you have given over the past years when some aspects of life got too much to process for me. My sincere thanks also to Smt. Shanthabai, Shri. C. H. Sreenivasamurthy and Smt. R. Sarojamma (my grandparents), Smt. Anuradha, Smt. Chandrika, Smt. Meenakshi, Smt. Sowbhagya (aunts), Shri. H. R. Sheshagiri, Shri. S. Sudheendranath, Shri. H. R. Anantharam, Shri. Badari Prasad and Late Shri. H. R. Vyasaraja (uncles), and my cousin brothers and sisters who have continually encouraged me in various ways through the different stages of my academic journey so far.

I also thank the family of my in-laws, Smt. Dakshayani, Shri. Nagesh Sastry, Mamatha, Saikrishna and Vaibhav for their trust and moral support through this journey. Finally, I thank my wife Sangeetha. Sangeetha, the past few years have been an intense roller-coaster ride. Thank you for your undeterred love, trust and belief in us that has helped us keep steady through this beautiful journey of life on the *pale blue dot*!

Mukunda Bharatheesha
Delft, June 2018

List of Publications

Journal Papers

W. J. Wolfslag, **M. Bharatheesha**, T. M. Moerland and M. Wisse
RRT-CoLearn: towards kinodynamic planning without numerical trajectory optimization,
IEEE Robotics and Automation Letters (RA-L): Accepted, (2018).

C. H. Corbato, **M. Bharatheesha**, J. A. van Egmond, J. Ju and M. Wisse
Integrating Different Levels of Automation: Lessons from Winning the Amazon Robotics Challenge 2016,
IEEE Transactions on Industrial Informatics: Special Issue on Recent Trends and Developments in Industry 4.0 Motivated Robotic Solutions, 2018.

Conference and Workshop Papers

R. B. Burger, **M. Bharatheesha**, M. van Eert and R. Babuška
Automated tuning and configuration of path planning algorithms,
2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 4371-4376 (2017).

M. Bharatheesha, N. Giftsun, C. H. Corbato, G. Dumonteil and M. Wisse
Dynamic Collision Avoidance for Collaborative Robot Applications,
IC^3-Industry of the future: Collaborative, Connected, Cognitive, Workshop at the IEEE International Conference on Robotics and Automation (ICRA), (2017).

M. Bharatheesha, and M. Wisse
Supervised Learning: A potential tool for feasible motion planning in State Space,
Beyond Geometric Constraints: Planning for Solving Complex Tasks, Reducing Uncertainty and Generating Informative Paths & policies: Workshop at the IEEE International Conference on Robotics and Automation (ICRA), (2015).

M. Bharatheesha, W. Caarls, W. J. Wolfslag and M. Wisse
Distance metric approximation for state-space RRTs using supervised learning,
2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 252-257 (2014).

Technical Report

M. Bharatheesha, R. B. Burger, M. de Vries, W. Ko and J. Tan
Motion Module for the Amazon Picking Challenge 2016 - Team Delft, RosCon 2016.

See Appendix C for information on author contributions in chapters with shared authorships.

About the Author

Mukunda Bharatheesha was born in Bengaluru, India, on 10th October 1984. He obtained a Bachelor of Engineering (B.E.) degree in the field of Electronics and Communications from Visveswaraya Technological University, Karnataka, India, in the year 2006. Subsequently, he worked for a period of 3 years in the field of embedded control software development for commercial and off-highway vehicles at Robert Bosch Engineering and Business Solutions Ltd., Bengaluru, India.

He then moved to The Netherlands in August 2009 to pursue his Master's education in Embedded Systems at Technische Universiteit Delft. He obtained his Master of Science (M.Sc) degree in August 2011. He conducted his M.Sc. thesis on the topic of Optimal Path Planning for Hopper Dredgers at IHC Systems B.V., The Netherlands. He was supervised by Prof. Dr. Ir. Robert Babuška. Following his M.Sc. graduation, he was a research assistant at the Delft Biorobotics Lab (DBL) for a period of 1 year where he worked on the autonomous navigation module for a custom designed differentially driven mobile robot, Robby, for the RoboCup@Home team of Delft.

He started his doctoral studies in October 2012 at the Delft Robotics Institute, TU Delft, in the field of sampling-based motion planning for robot manipulators. He was supervised by Prof. Dr. Ir. Martijn Wisse from the Robot Dynamics group. A part of his doctoral study involved the development and integration of motion planning modules for collaborative robotic systems in the EU project Factory in a Day. Between May and November 2014, he was a visiting researcher at LAAS-CNRS, Toulouse, France, where he worked with Dr. Florent Lamiraux of the Gepetto team. During the course of his doctoral studies he has supervised multiple B.Sc and M.Sc theses projects and also assisted in teaching activities for the M.Sc course Humanoid Robots. He also lead the development of the motion planning module for the double world championship winning Team Delft at the Amazon Robotics Challenge 2016.

His current research activities involve the study of effectively combining supervised learning and optimal control for dynamically constrained motion planning with sampling-based planners. He is also currently leading the development of a Massive Open Online Course (MOOC) on the fundamentals of Robot Operating System (ROS) scheduled to start from September 2018 on the edX online education platform.

Propositions

accompanying the dissertation

SAMPLING-BASED MOTION PLANNING IN CONFIGURATION AND STATE SPACES
USING SUPERVISED LEARNING TOOLS

by

Mukunda BHARATHEESHA

1. Of all the existing motion planners, RRT-Connect with (random) short-cutting has the best chance of making it to an industrial robot teach pendant.
2. Translational redundancy in a bin picking robotic system significantly reduces the *randomness* of solutions from randomized planners. (Chapter 3)
3. It is rather euphemistic to call a robot *collaborative* if it only stops moving *after* colliding with a human. (Chapter 4)
4. Supervised learning-based approximations of the optimal cost-to-go in state space is about three orders of magnitude faster than solving for it. (Chapter 5)
5. Learning-based RRTs outperform state space RRTs using LQR heuristics in generating motions that effectively utilize *natural* dynamics.
6. An end-user usually decides on the *quality* of a motion planning algorithm based on the *default* parameter settings of its software implementation.
7. If *all* users of open source robotics software contribute back to its betterment, robotics research will see tremendous progress.
8. The apprehensions about robot cognition will prevent its actual realization.
9. Governments should reserve a portion of the revenue from robotization for socio-economic welfare.
10. Honesty and integrity are nodes and edges of the graph of happy life and we actually learn the costs of those edges along the way.

These propositions are regarded as opposable and defendable, and have been approved as such by the supervisor prof. dr. ir. M. Wisse.

Stellingen

behorende bij het proefschrift

SAMPLING-BASED MOTION PLANNING IN CONFIGURATION AND STATE SPACES
USING SUPERVISED LEARNING TOOLS

door

Mukunda BHARATHEESHA

1. RRT-Connect met "random shortcutting" heeft van alle beschikbare padplanningsalgoritmen de hoogste kans om ooit te verschijnen op het bedieningspaneel van industriële robots.
2. Een redundante lineaire bewegingsvrijheid in een "bin picking" applicatie vermindert de *willekeurigheid* van oplossingen van randomized planners.
3. Een robot *collaboratief* noemen indien deze slechts stopt nádat er een botsing plaatsgevonden heeft is nogal eufemistisch.
4. De benaderingen van het "optimal cost-to-go"-probleem in state space waarmee een "supervised learning"-aanpak aankomt zijn ongeveer drie ordes van grootte sneller dan werkelijke oplossingen voor dit probleem.
5. Lerende RRTs genereren betere paden dan state space RRTs wanneer de eersten LQR heuristieken bevatten die gebruik kunnen maken van *natuurlijke* dynamiek.
6. De meeste eindgebruikers beoordelen padplanningsalgorithmen enkel op de resultaten die deze algoritmen behalen bij gebruik van de *standaardinstellingen*.
7. De ontwikkeling van open-source roboticasoftware zou significant sneller gaan als *al* haar gebruikers hieraan zouden bijdragen.
8. *Werkelijk* cognitieve robots zullen door de angst hiervoor (zeer waarschijnlijk) nooit gerealiseerd worden.
9. Een deel van de inkomsten verworven met commerciële robotica-activiteiten zou door overheden aangewend moeten worden voor het onderhouden van het sociale stelsel.
10. Eerlijkheid en integriteit zijn de knopen en zijden van de graaf van een gelukkig leven en ervaring laat ons de gewichten aan deze zijden gedurende ons leven toekennen.

Deze stellingen worden opponeerbaar en verdedigbaar geacht en zijn als zodanig goedgekeurd door de promotor prof. dr. ir. M. Wisse.