Lehrstuhl für Informationstechnische Regelung
Technische Universität München
Univ.-Prof. Dr.-Ing. Sandra Hirche

Real-time Robotic Motion Control and Adaptation in Constrained Environments

Thomas Julian Nierhoff

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. G. Cheng, Ph.D.

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. S. Hirche
2. Prof. Dr. Y. Nakamura
   University of Tokyo / Japan

Die Dissertation wurde am 27.01.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 09.09.2015 angenommen.

Abstract

Technological advances of robotic systems lead to a paradigm change of their role in society. Whereas in the past robots were mostly used for repetitive tasks in industry, they are bridging the gap these days into everyone's life, fulfilling a diversity of complex daily tasks. Different from automation tasks in a factory, the environment is often unstructured and not known a priori in such a situation. This imposes additional challenges on how a robotic motion is executed best. When facing tight time constraints - e.g. during interaction with humans - the classical way of calculating an robotic motion without prior knowledge becomes quickly unfeasible in a complex environment. An alternative approach is *Programming by Demonstration* where a optimal prototypic motion is learned beforehand for every task and only adapted to fit environmental constraints. Low computational complexity and a small deviation from the demonstrated movement in the presence of disturbances and obstacles are two key features that must be maintained during adaptation for smooth real-time execution and the recognition of the original motion. Existing schemes are either learning-based and thus computationally expensive or possess only minor adaptation capabilities.

The contribution of this thesis is a computationally efficient approach to reproduce and adapt a demonstrated motion with a (humanoid) robot. By using a least-squares optimization, additional constraints imposed either by another interaction partner or the environment can be considered in addition to the local motion properties and solved in closed form. The framework is extended to handle generic robotic problems including collision avoidance in highly constrained environments, cooperative manipulation of two or more robots and efficient collision checking methods. A set of controllers with similar optimality properties is derived, bridging the gap between planning and control and allowing to suppress disturbances in real-time while maintaining the original motion shape. Simulations and experiments with three different robotic platforms verify its applicability to a variety of tasks under real-time constraints.

Special focus is drawn on a holistic approach to keep the shape of the resulting motion consistent in the three domains of planning, control and reasoning. Different from conventional approaches which favor a strict separation, this work aims at a tight coupling to keep the optimality properties consistent for the entire process chain.

Zusammenfassung

Fortschreitende technische Entwicklungen auf dem Gebiet der Robotik führen zu einem Paradigmenwechsel in Bezug auf deren Platz in der Gesellschaft. Im Gegensatz zu Industrierobotern mit standardisierten Bewegungsabläufen sind heutige Systeme in der Lage, eine Vielzahl komplexer Aufgabenstellungen des täglichen Lebens zu bewältigen. Im Gegensatz zu herkömmlichen Automatisierungsaufgaben ist das Umfeld hierbei oftmals unstrukturiert und nicht im Vornherein bekannt, was zu zusätzlichen Herausforderungen bezüglich einer optimalen Roboterbewegung führt. Sind Echtzeitbedingungen in komplexen Umgebungen gefordert - z.B. bei der Mensch-Roboter-Interaktion - erreichen klassische Bewegungsplaner schnell ihre Grenzen. Ein alternativer Ansatz ist *Programming by Demonstration*, bei dem eine optimale prototypische Bewegung für jede Aufgabe bereits im Voraus gelernt wird und während der Ausführung nur noch angepasst werden muss, um Umgebungsbeschränkungen zu genügen. Für eine flüssige Ausführung ist einerseits ein geringer Rechenaufwand notwendig, auf der anderen Seite darf die resultierende Bewegung selbst unter dem Einfluss von Störungen und Hindernissen nur eine geringe Abweichung von der demonstrierten Bewegung aufweisen, um als solche erkannt zu werden. Bestehende Ansätze sind entweder learning-basiert und daher rechenintensiv oder besitzen nur geringe Anpassungsfähigkeiten.

Der Beitrag dieser Dissertation ist ein rechnereffizienter Ansatz, um eine vorab demonstrierte Bewegung mit einem (humanoiden) Roboter wiederzugeben und anzupassen. Mithilfe eines least-squares Ansatzes können sowohl die lokalen Merkmale der Bewegung als auch zusätzliche umgebungs- oder interaktionsspezifische Beschränkungen beschrieben und in geschlossener Form gelöst werden. Mehrere Erweiterungen für Kollisionsvermeidung in verblockten Umgebungen, kooperative Manipulation mit zwei oder mehr Robotern und effiziente Kollisionserkennungsmethoden machen den Ansatz für eine Vielzahl an typischen Robotikaufgaben tauglich. Die Herleitung von Reglern mit den gleichen unterlegten Optimalitätseigenschaften schliesst die Lücke zwischen Planung und Regelung und erlaubt die Unterdrückung von Störungen in Echtzeit unter gleichzeitiger Beibehaltung der originalen Bewegungseigenschaften. Die Validierung des Ansatzes für eine Vielzahl an Aufgaben unter Echtzeitbeschränkungen erfolgt sowohl durch Simulationen als auch mittels Experimenten unter Einbeziehung von drei verschiedenen Roboterplattformen.

Ein Schwerpunkt dieser Arbeit liegt auf einer ganzheitlichen Betrachtung, um die Form der resultierenden Bewegung in den drei Domänen Planung, Regelung und Schlussfolgerung konsistent zu halten. Im Unterschied zu konventionellen Ansätzen, welche eine strikte Trennung favorisieren, zielt diese Arbeit auf eine enge Kopplung, um die Optimalitätseigenschaften während der gesamten Wirkkette beizubehalten.

Contents

Notations

Abbreviations

2D	two-dimensional
3D	three-dimensional
6D	six-dimensional
AAD	Allowed angular deviation
ARAP	As-rigid-as-possible
COM	Center of mass
CHMM	Continuous Hidden Markov model
DLS	Damped least squares
DOF	Degree of freedom
DMP	Dynamic movement primitive
DTW	Dynamic time warping
EE	Endeffector
FFT	Fast Fourier transform
GMM	Gaussian Mixture model
GMR	Gaussian Mixture regression
HMM	Hidden Markov model
IK	Inverse kinematics
ILQR	Iterative linear quadratic regulator
LS	Least squares
LQR	Linear quadratic regulator
LTE	Laplacian trajectory editing
MDP	Markov decision process
MLE	Maximum likelihood estimation
PCA	Principal component analysis
PL	Piecewise linear
PbD	Programming by demonstration
RLS	Recursive least squares
RRT	Rapidly-exploring random tree
SMA	Spatial moving average
SR	Singular-robust
SVD	Singular value decomposition
WDLS	Weighted damped least squares

Conventions

x	Scalar
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$\mathbf{X}^T$	Transposed of $\mathbf{X}$
$\mathbf{X}^{-1}$	Inverse of $\mathbf{X}$
$\mathbf{X}^+$	Pseudoinverse of $\mathbf{X}$
$\mathbf{X}^\#$	SR-Inverse of $\mathbf{X}$
$\|\|\cdot\|\|_p$	p-norm
$\|\cdot\|$	Absolute value
$X \bigoplus Y$	Minkowski sum
conv(x)	convex hull
det($\mathbf{X}$)	determinant of $\mathbf{X}$
rank($\mathbf{X}$)	rank of $\mathbf{X}$
$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$	partial derivative of $f(\mathbf{x})$
$\mathbf{X}_{\{3:\}}$	third row, entire column of $\mathbf{X}$

Main variables

Motion adaption and planning

t	time
$\mathbf{p}$	discrete sampling point
$\mathbf{P}$	discrete path
δ	local path property
$\mathbf{L}$, $\mathbf{\Delta}$	local path property matrices
$\bar{\mathbf{P}}$, $\bar{\mathbf{C}}$	constraint matrices
ω	weighting factor
$\mathbf{\Omega}$	weighting matrix
$\mathbf{R}$	rotation matrix
E	similarity measure

Motion imitation and control

$\mathbf{p}$	discrete sampling point
$\mathbf{P}$	discrete path
$\mathbf{x}$	task space coordinates
$\mathbf{q}$	joint space coordinates
$\mathbf{u}$	control input
c	cost function
$\mathbf{J}$	Jacobian matrix
w	weighting factor
$\mathbf{W}$	weighting matrix
$\mathbf{A}$	system matrix
$\mathbf{B}$	input matrix
$\mathbf{K}$	LQR gain
S	LQR cost function
$\mathbf{V}$	LQR cost function matrix
$\mathbf{Q}, \mathbf{R}, \mathbf{L}, \mathbf{M}$	LQR weighting matrices
$\mathbf{\Sigma}$	covariance matrix
$\mathbf{d}$	distance vector

Motion reasoning

S	state
A	action
T	transition function
R	action reward
V	value function
Q	action-value function
π	policy
p	stroke impulse
θ	stroke angle
κ	subjective difficulty
P_o	objective difficulty
γ	discount factor

List of Figures

1

Introduction

This chapter familiarizes the reader with the general topic of the thesis, its application domains and properties. Starting with 1.1, the general principles of movement generation and adaptation are recapitulated. When applied to robots, this leads to a set of challenges which are identified and characterized in Sec. 1.2. An overview of how the challenges are tackled is given in Sec. 1.3, corresponding to the outline of this thesis and its main contributions.

1.1 Basics of motion imitation, adaptation and reasoning

Nature features a rich variety of different movements amongst their inhabitants, many of them which still look like a miracle regarding speed, complexity or simplicity. Mantis shrimps are able to accelerate theirs claws at an astonishing acceleration of more than $10000g$ when hunting preys [241], elephants have to coordinate around $40000-50000$ muscles when moving their trunk [273] and the flocking motion of bird swarms is explained by a set of remarkable basic rules [252]. Humans are fascinated by these behaviors for a long time, trying not only to understand but also to reproduce them. A proof of the early efforts are mechanical automata, which date back to the 16th century when the robotic monk was manufactured, capable of walking around, lifting his cross and remain in silence while praying [96]. Other similar works include Tipu's tiger [286] or the digesting duck [315], testifying the impressive craftsmanship at that time. Even if these automata had an ingenious mechanical movement mechanism, they could only reproduce a very limited set of motions. On the other hand animals and humans can not only execute a variety of motion types but also modify them depending on the context. Whereas the motion of walking is of rather general scope, every human shows a different

walking style depending on the texture of the surface he walks on, its inclination or the walking speed. They are also able to learn new types of movements throughout their life, e.g. when playing a new type of sports or dealing with physical impairments like an implant or amputation. All those effects are investigated in the discipline of motor skill, trying to answer the question how an intentional movement is learned and executed to fulfill a goal-oriented task [43, 215, 283].

Despite all technological advances, such capabilities are still limited to living creatures. Yet there is a huge branch in the field of biomimetics and bioinspiration to take advantage of useful developments in nature and synthesize or adapt them to solve human-related problems. Prominent examples are velcro fasteners inspired by burs sticking to fur and clothes or swimsuits with a surface texture in the style of a sharkskin for lower drag when moving through water. The highly specialized figure of certain animals has motivated researches to build robots based on the same design principles in order to exploit their capabilities and imitate their style of movement. The sticky surface of gecko toes is used as inspiration for StickyBot [62], a mechanical climbing robot using synthetic setae and comparable shape to a real gecko. Other robots resemble the shape and motion of snakes, consisting of a single hyperredundant actuator that can travel along narrow pipes and imitate a sidewinder's motion [79, 214]. Regarding human motion and appearance there exist multiple humanoid robots like Honda's ASIMO [119] or the different HRP platforms [138, 139]. As all of them are able to interact with humans, they support the trend from humans and robots working in independent domains like a fully automatized assembly line towards dynamic and interactive coexistence [64, 72, 236, 254, 299]. Whereas they are a promising approach for being accepted as a human-like interaction partner [153, 204] they also show the limitations of today's technology as a humanoid robot with similar skills and abilities as a real human is not yet in sight. Besides technological challenges regarding miniaturization and a only superficial understanding of how cognition works [2], this is also due to a missing knowledge about how to generate suitable motions which are on the one side able to fulfill a given task but also pleasant to humans and capable of sufficing environmental constraints. Especially in the context of human-robot-interaction it is this unique combination of all three factors which makes movement generation challenging.

Caused by these new demands and increasingly sophisticated robot designs engineers are reconsidering how to let the robot execute a given motion. To generate a new robotic movement the standard way was for a long time to program every single movement step by step. This method becomes strenuous for complex movement or highly redundant manipulators as it requires multiple iterations of refinement and adaptation. Another option is motion imitation. By having a human expert at hand who demonstrates the desired motion to the robot, mapping algorithms [199, 240, 290] or kinesthetic teaching [3, 116, 163, 184] let the robot move in a similar way. The movement is then recorded for latter manual adjustment or multiple reproductions. Differing from the standard way where complexity scales with the length of the movement and the number of robot joints, the motion imitation approach is very simple, fast and can be redone easily once the mapping algorithms have been developed. Implicitly it is assumed that the expert executes an optimal motion, omitting the need to find the underlying optimality principle behind

each motion manually.

Once encoded and transferred to the robot, it imitates the specific motion. Yet it is unable to fit the given motion to a new environment, a changed task or robot-specific constraints occurring during reproduction. Such constraints include joint angle limits or balance of the robot [4, 319]. A motion adaptation scheme is necessary that suffices additional constraints while keeping overall deviation from the original motion small, see Fig. 1.1. Without adaptation scheme, there is no way of modifying a imitated motion and thus every different motion requires a new teaching. The used adaptation scheme also influences the amount of adaptation without creating degenerated motions. Among the most general and robust methods are explicit planning methods [180] and optimization routines [98, 274, 287, 303] and learning methods [108, 125] yet they are slow and require an offline computation. Another option are online adaptation methods [150, 298, 321], but they lack the flexibility of the explicit optimization methods. It is thus a still ongoing challenge to find adaptation schemes which are both general and fast. Motion imitation and motion adaptation are closely related to motion reasoning. When

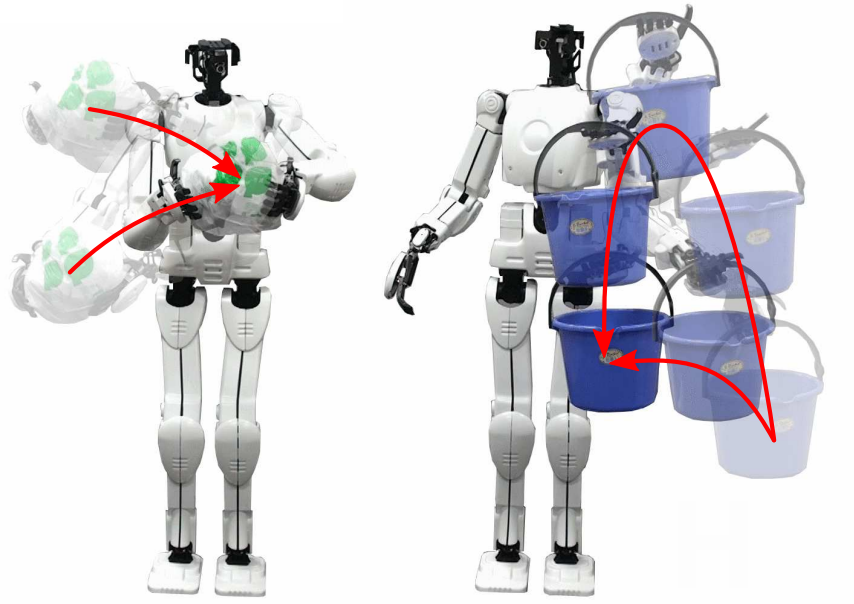


Figure 1.1: Motion adaptation with a humanoid robot. Left: Adaptation to a different start position. Right: Adaptation to a different motion shape

adapting a recorded motion, an infinite variety of different ways exist. Yet for a given task some adaptation schemes are better than others. A suitable motion reasoning scheme has to be aware of the current situation, judge it and select best-fitting adaptation scheme for the given task [8, 172, 211]. Multiple aspects have to be considered such as the used representation for motions, decision rules, the temporal dependencies between multiple motions and the immediate and long-term effect of each motion [26, 97, 165].

The application domain of a (humanoid) robot capable of adapting learned motions is manifold, see [1]. They expect an increasing demand of robotic technology in non-manufacturing areas like healthcare, civil service, commercial and consumer market. One

goal is to replace existing human workforce when needed, for example if disasters and accidents do not allow humans to enter endangered zones themselves [216] or because the task would be too bothersome for elderly or handicapped people [233]. When equipped with advanced sensing and motor skills, a humanoid robot can operate human-tailored devices (automats, cars, portable electric/electronic devices) in a straightforward fashion [232], making it capable of fulfilling daily life human tasks. For interaction with humans in society, observations in the context of human-robot-interaction indicate human-like motions increase both the acceptance of (humanoid) robots as interaction partners and their predictability [99, 166, 173]. Building upon results covered in this thesis, a cognitive framework may allow robots in the future to autonomously acquire task knowledge by looking at other's motions and adapt them for their own purpose. This will enable virtually everyone to serve as a robot programmer when needed by simply demonstrating the desired task to the robot.

1.2 Challenges in motion imitation and adaptation

Different from conventional motion planning schemes there are a number of specific challenges associated with robotic motion imitation and adaptation. To make these challenges mathematically tractable, the entire movement generation process is represented as a minimization problem of the general form

$$\begin{aligned} &\text{optimize } g(x, \hat{x}), \\ &\text{s.t. } h(u). \end{aligned} \tag{1.1}$$

In the optimization problem above, $\hat{x}$ and x stand for the demonstrated and adapted motion. Multiple motion representations are possible, for example a deterministic way using a n-dimensional trajectory or vector field. Another option is a probabilistic encoding through hidden Markov models (HMM) or Gaussian mixture models (GMM). Due to the variety of motions, no further assumtions are made, explicitly ignoring time-dependency (optional but not necessary) or the underlying dynamics of the motion (for slow motions the kinematics are sufficient as well). Based on the chosen representation, the function $g(x, \hat{x})$ calculates a distance between the two motions, indicating how much they differ from each other. Common measure include the spatial distance, represented by the norm of a vector valued function or the likelihood if a probabilistic encoding is used. If both motions x and $\hat{x}$ differ, the goal becomes to find an optimal value for $g(x, \hat{x})$. The variable u serves as an abstract representation for the given task which may be described by "bring me a cup of coffee" or "'wash the laundry". The function $h(u)$ then translates the task into suitable boundary constraints that have to be fulfilled in order to complete the task. Note that $h(u)$ specifies the type of constraint independent of the specific motions $\hat{x}$ or x which are hence no function parameters of $h(u)$. Still the constraints imposed by $h(u)$ must be in the same domain as the motion to be constrained.

Reducing both motion imitation and motion adaptation to an optimization problem has several advantages. When facing the decision which imitated motion to choose from a set of possible options, the cost function $g(x, \hat{x})$ provides an intrinsic measure to evaluate

the quality of each motion. In this sense it acts as an automated decision maker for the human that also fits the desired movement to the constraints in $h(u)$ in an optimal way. As there are highly optimized algorithms for a vast class of optimization problems available today, it also results in a faster approach compared to manual evaluation and adaptation. Still these algorithms only provide an optimized solution that is as close as possible to the true solution. If the imposed constraints are conflicting, the optimization result can be arbitrarily bad.

Regarding the differentiation of motion imitation, adaptation and reasoning, motion imitation is about finding viable transformation schemes to transfer a recorded motion to the robot which results in $\hat{x}$. It also covers the task of finding a proper representation for $\hat{x}$ that lets the robot imitate a given task. For example, if someone has to carry a box by hand, recording and reproducing only the left foot's position will most likely result in an unsuccessful task completion. For a better representation scheme it is necessary to consider also the position and orientation of both hands. After a movement representa-

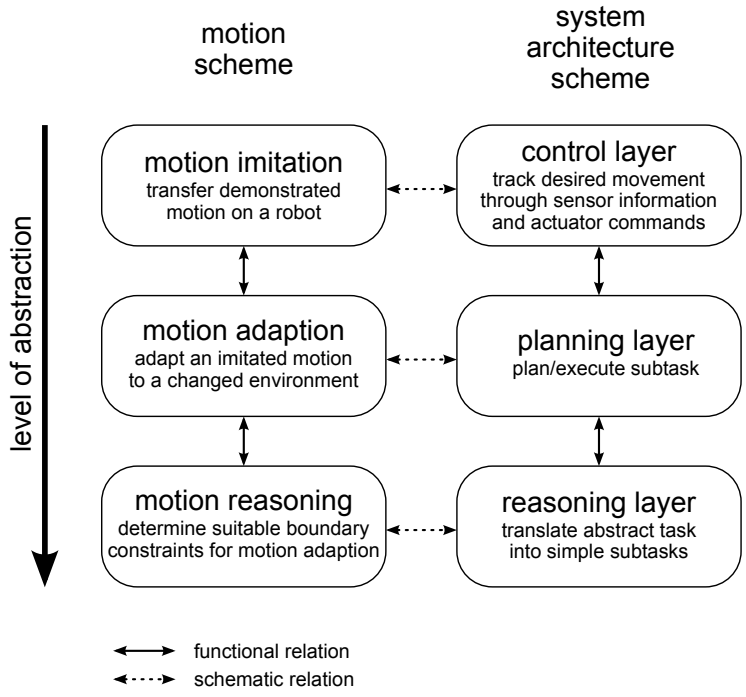


Figure 1.2: Functional and schematic relation between different motion schemes and corresponding system architecture schemes

tion scheme is determined, boundary constraints $h(u)$ must be determined by a motion reasoning algorithm that perform the task successfully in a different environment. For the task "carry the box by hand" this can be the finding that the box has to be held in an upright position while grasping it firmly with both hands, corresponding to the underlying optimality criterion of the task. Once both boundary constraints and a proper motion representation are given, the motion adaptation scheme has to solve (1.1) while taking care of the underlying robot-specific constraints. Among the most common ones are dynamic constraints (the robot can only achieve a certain velocity/acceleration with-

out damaging gears and motors) and kinematic constraints (certain robot poses result in collisions with the robot itself or objects in the environment).

The challenges associated with such an approach are twofold. The first problem arises from the interaction between the three schemes of motion imitation, adaptation and reasoning, requiring good mathematical methods and a proper framework for real-time execution on a robot. The authors of [164] suggest using a three-layered architecture, which we redefine as a control, planning and reasoning layer. The reasoning layer corresponds to motion reasoning as it translates an abstract task for a given state into simpler subtasks which can be processed by the planning layer. The planning layer then calculates a suitable movement which is tracked by the control layer by means of the available sensor information and actuator commands. Both planning and control layer show a tight correlation to the motion adaptation and imitation scheme. On the one hand planning a motion is often rewritten as optimizing a function with additional boundary constraints. On the other hand the control layer only tracks information encoded in $\hat{x}$ and x, this however depends on the used representation for motion imitation.

The second challenge is consistency. For a seamless interaction between all three layers, they must not contradict with each other as otherwise the demonstrated movement cannot be adapted. Ideally they are based on the same optimality principles. To illustrate the problem, we consider again the "carry the box by hand"-scenario. If there are fragile items in the box, it is not sufficient anymore just to carry the box in an upright position. One must also minimize the acceleration exerted on the box as not to break objects inside the box accidently. This new aspect requires modifications of all layers: The reasoning layer has to split the general task into subtasks that can be fulfilled by the planning layers without exceeding a certain acceleration, the planning layer has to calculate a feasible motion and the control layer must track the desired motion while considering not only positional offsets but also the amount of acceleration. Keeping all three layers consistent is a challenging task, especially if hard constraints must be met during adaptation. If the optimality criterion is complex, planning a motion or tracking it in real-time might not be possible. In a similar manner it can take too long for the reasoning layer to find an optimal command to be executed by the planning layer, for example when trying to calculate an optimal next move for a game of Go. In this case an approximate optimality criterion has to be found, which resembles the true criterion sufficiently well to complete the task in an adequate manner.

1.3 Contributions and outline of the thesis

This thesis aims at a holistic approach for real-time adaptation of an imitated motion from an expert demonstration to different tasks and changed environmental situations. It pursues an trajectory-driven approach in which both the demonstrated and the adapted movement are encoded as discrete trajectories. This representation allows to tackle various application-oriented questions regarding the feasible amount of adaptation, different ways of dealing with hard environmental constraints, tight real-time constraints or a consistent planning and control architecture. Following the partitioning described in the last section, the presented methods are grouped in three chapters, corresponding to the con-

trol, planning and execution layers in Fig. 1.2.

Chapter 2: Planning

Chapter 2 tackles the challenge of real-time motion planning in heavily cluttered environments while preserving the original motion shape. To allow for fast modifications and provide a good flexibility, a least-squares based optimization approach is presented. This provides a closed form solution and allows the prioritization of additional constraints. Inspired by the field of computer graphics, the discrete Laplace-Beltrami operator is used to encode the trajectory properties on a local scale and maintain them during adaptation. The chapter focuses on three main aspects of the motion adaptation process which are essential for generic trajectory modification in cluttered environments: A low computational complexity, guaranteed spatial bounds and efficient approaches to avoid obstacles during adaptation. It is shown how the computational complexity can be reduced through a hierarchical approach which downsamples the trajectory before deformation or an alternative representation using splines that can be calculated efficiently in the continuous domain. Depending on the used approach, spatial bounds allow for a fast feasibility check of the adaptation process which further reduces computational complexity. To account for spatial obstacles in the environment, three different collision avoidance methods are presented. For mostly unconstrained environments it is sufficient to rely on spatial bounds to avoid obstacles. If these spatial bounds become too conservative, a reactive collision avoidance scheme circumnavigates obstacles through repellent force fields. In heavily constrained environments, a sampling-based approach pursues a different idea by not deforming a given trajectory but rather growing a search tree with optimal local tree properties while circumnavigating obstacles. The results in this chapter and the related appendix are either partly or in their entity based on the author's work in [223, 225, 226, 228].

Chapter 3: Control

The challenge of developing a controller that is consistent in its underlying optimality criterion with a given motion planning scheme is addressed in chapter 3. To ensure safe interaction with the robot, stability must be guaranteed in addition. Caused by the interlocking nature of the local trajectory properties it is necessary to consider not only the current state but also future states when developing an optimal controller. To account for both aspects the underlying cost function of the planning framework is embedded in a linear quadratic regulator (LQR) controller, resulting in a predictive control scheme with guaranteed asymptotic convergence. Kinematic singularities are avoided by extending the approach to follow a given reference trajectory both in joint- and task space. It turns out that the method resembles a predictive version of the weighted damped least squares (WDLS) inverse kinematics approach. As the used motion representation influences the motion adaptation process to a large extent, two other common representations

are presented and evaluated for their tracking capabilities. Whereas the first, vector-field method using a set of piecewise linear functions is mainly used for endeffector (EE) control, a HMM based approach is applicable to close-contact scenarios while taking into account the full-body motion of a humanoid robot. The results in this chapter and the related appendix are either partly or in their entity based on the author's work in [224, 227, 229].

Chapter 4: Reasoning

Chapter 4 addresses the problem of pursuing an abstract task through decomposition into smaller subtasks which can be processed by an underlying motion planning framework. An Markov decision process (MDP) representation allows to consider a variety of effects while deriving policies to let the robot act in an optimal manner. Being a model-based approach, it relies on a precise model of all environmental effects. This becomes challenging if difficult processes like human behavior must be modeled. Pool is used as a test bed to evaluate all effects and measure their influence. When facing a human with a robot, a dynamic programming framework calculates an optimal robot motion for each move, depending not only on the robot's capabilities but also on the opponent. By learning the opponent's behavior and adapting the motion of the robot to counter him, a significant improvement is achieved. The results in this chapter and the related appendix are either partly or in their entity based on the author's work in [188, 222, 230].

2

Motion adaptation and planning

This chapter focuses on motion adaptation of a previously demonstrated movement. Differing from existing state-of-the-art-approaches, it allows to encode and adapt a demonstrated discretized trajectory in real-time while preserving its local features. The presented Laplacian trajectory editing (LTE) framework is based upon a least squares (LS) approach to provide an optimal weighting between the local trajectory properties and additional constraints. Due to its generality it serves as a common basis for further extensions. The key finding of this chapter is to show how LTE can be used to fulfill a variety of typical robotic problems with only minor modifications. Depending on the complexity of the problem, the spectrum of applications ranges from real-time deformation of a given trajectory to explicit offline planning of an optimal path in a labyrinth-like environment.

The chapter is organized as follows: In Sec. 2.4 LTE is introduced as the basic formulation of the motion adaptation framework throughout this thesis. Sec. 2.5 discusses various properties of the method, not only highlighting its capabilities but also showing its drawbacks and how they are overcome. The multiresolution approach in Sec. 2.7 provides an computationally efficient way of deriving a solution while also considering nonlinear deformation effects. A cooperative manipulation scheme in Sec. 2.8 extends the approach to more than one robot in order to manipulate objects in a synchronized manner. Spatial bounds of the approach are derived in Sec. 2.6, allowing a fast approximate solution to be estimated a priori without the need to calculate an explicit solution. Last, Sec. 2.9 focuses on collision avoidance during the adaptation process if not only free-space trajectories are considered. Most presented approaches are evaluated not just through simulations but also real life experiments in Sec. 2.10. A summary in Sec. 2.11 quickly reviews the general aspects of LTE and bibliographical notes in Sec. 2.12 highlight the relation to previously published papers about the topic.

2.1 Introduction

Humans are able to generate and adapt a vast variety of goal-driven movements without apparent effort. Scientists have developed different theories to address this degrees of freedom problem in motor control [29, 110, 262, 301, 314], still there is no unifying and profound theory at hand. This is problematic as more and more sophisticated robots are available today, with a comparable number of degree of freedom (DOF) as the human body. In this context the hardware seems to be ahead of software. Although there exists a vast number of different robots which vary in shape, functionality and complexity [55, 92, 205], most of them are only capable of performing very primitive and highly specific movements.

It is still unknown when or even whether the human movement mechanisms can be decoded in their entity. Until then, the process of human movement generation must be seen as a black box, with sensors perceiving the environment, an unknown information processing structure and whole-body movements as the resulting action. This knowledge is exploited to a certain extent by roboticists. Even if the information processing structure is unknown, it is possible to observe and measure the human movement for a given environment and imitate the movement with a robot [156, 160] in order to perform human-like motions and facilitate the interaction with humans [153, 154, 173, 237].

For a humanoid robot and a recorded whole-body movement, the movement is imitated in its entity. Other researchers focus only on specific parts of the entire body and map the movement to a robot with completely different structure. A good movement imitation scheme allows the successful movement reproduction in a similar environment. This is only of limited practicability for real problems as even small modifications require an adaptation of the reproduced movement. Unless aforementioned information processing structure is known, it has to be mimicked manually through optimization of a proper optimality principle.

So far this approach is entirely human-driven as it tries to resemble and understand a human motion as good as possible [212, 284]. Another aspect comes from engineering as demonstrating a motion to a robot is an efficient and fast method for executing a complex motion on a robot with multiple DOF. Once an approximate motion is found, it must be fit to the task through a proper motion adaptation scheme. Usually this motion adaptation scheme cannot be defined arbitrary but has to meet specific criteria regarding smoothness or collision avoidance.

2.2 Related work

Motion adaptation as defined before belongs to the wider class of programming by demonstration (PbD) where a new robotic movement is not programmed from scratch but based on a previously demonstrated movement. Whereas first approaches date back to the 90's [95] it gained widespread attention just during the last decade [34]. PbD approaches have different requirements than conventional motion generation schemes. As the movement generation process with PbD is either partially or fully automatic, the

type of movement encoding limits the adaptation capabilities. Depending on the requirements, different encoding schemes are used. Among the most common ones is GMR, encoding a motion by a set of Gaussian kernels with different shape and weight [50, 51, 320]. The method has been extended to account for multiple constraints both in joint and task space during adaptation [47]. In addition the combination with HMMs increased robustness towards temporal deviations [49]. Another option are dynamic movement primitives (DMP) [82, 260, 261], modulating a second order point attractor to fulfill additional constraints like collision avoidance or variable goals [121, 238]. Both methods encode the resulting prototypic trajectory through a highly reduced number of variables, thus easing the adaptation process at the cost of a large computational complexity during the learning/encoding step.

Another option are adaptation schemes working either directly on trajectories or using an encoding that does not need to be learned. Prominent examples are the Elastic Strips framework for obstacle avoidance while maintaining a shortest distance path [39], Elastic Bands for minimal deformation in joint space [250], trajectory retargeting maintaining affine features [244, 245] and reactive collision avoidance schemes through repellent force fields [20, 152, 175, 253] or velocity obstacles [89].

One can also use the intrinsic smoothness of specific trajectory representations based on splines [178] or Bézier curves [117]. All methods described so far have in common that they are of rather limited scope, either allowing trajectory modifications only in the space they are defined or working well only in conservative environments. Whereas this is often associated with a low computational complexity during adaptation, it also limits the applicability if complex dynamical constraints have to be considered or the surrounding is cluttered and heavily unorganized. In the latter case an explicit numeric optimization is often the only viable option, shifting focus from online adaptation towards planning. Two different classes exist: Direct adaptation methods and indirect adaptation methods. Whereas direct adaptation methods modify the motion directly based on the constraints in task space, indirect methods depend on a cost function calculated from a set of demonstrations and valid over the entire task space [189, 206, 207]. Even if indirect methods can have better generalization capabilities, the cost function is difficult to obtain for complex movements and provides a good approximation to the true cost function only on a local scale, requiring multiple repetitions to cover the entire task space. Direct methods are easier to handle as they allow the user to modify the cost function directly by imposing additional constraints with corresponding weights. When used with PbD, the similarity between original and adapted motion is encoded in the constraints [137, 263, 326].

If the environment is even more constrained, aforementioned methods are prone to fail. Yet by giving up the similarity constraints a solution may still be found. In this case we are focusing on pure path/motion planning. With increasing computational power sampling-based motion planners are nowadays the most prominent approach, exploring the entire search space randomly until a valid solution is found. Such methods include probabilistic roadmaps [122, 147], rapidly exploring random trees (RRT) [127, 134, 179, 181, 187] and RRT*, a modified version of RRT with global optimality properties [140, 142, 147, 179, 243].

2.3 Problem statement and conceptual approach

Recapitulating the adaptation problem in (1.1), the goal of motion adaptation is to find a solution for

$$\begin{aligned} &\text{optimize } g(x, \hat{x}), \\ &\text{s.t. } h(u), \end{aligned} \tag{2.1}$$

such that the adapted motion x matches the original motion $\hat{x}$ best while sufficing additional constraints $h(u)$. It turns out that all PbD approaches presented in the previous section can be described by (2.1) even if their exact definition differs from approach to approach. Besides the type of encoding, additional design criteria must be considered.

One challenge is computational complexity. It covers two aspects, execution time for realistic applications and theoretical scaling effects investigating the asymptotic behavior. A low execution time has several advantages as it is not limited to an a priori offline computation but allows for an online replanning to react to sudden disturbances. In the ideal case, the method is fast enough to calculate multiple adaptation strategies in real-time [93]. Computational complexity is intrinsically related to the underlying mathematical optimization problem, leading to specific optimization algorithms that must be used. As nonlinear optimization techniques are susceptible for local minima, finding the global minimum is a time-consuming undertaking and thus not applicable. A special class are convex optimization problems as they have only one unique optimal solution. Once an optimum is found, it is known to be the global optimum. Sadly no analytical form exist in general, requiring iterative optimization algorithms which are computationally demanding [263]. This drawback is overcome by linear LS [101] as for any problem formulated by a set of linear equations, there is a unique LS solution fulfilling the equations in an optimal manner.

Another challenge are generalization capabilities, referring to the general nature of the underlying mathematical framework. At this point one is often forcing a tradeoff between a general yet unspecific approach and an approach explicitly designed for special needs yet impossible to be applied directly to solve a related problem. On the other hand it aims at how easily the approach can be extended to fit additional needs. The two aspects are related as even a highly specific approach can be easily expanded to fit a wider class of problems. In a similar way, expansions make an otherwise highly unspecific approach suitable for a set of very special problems. When expanding a given approach one can either choose a bottom-up or top-down approach. The bottom-up approach looks at a highly specific problem with is further generalized and abstracted afterward through suitable expansions. In contrast the top-down approach first tackles a rather generic and application-unspecific problem, driven for example by an abstract mathematical problem. Either through proper interpretation or modifications of the original approach it is then used to solve user-specific problems. As such GMR as in [46] is a special case of Bayesian regression [268] used successfully to classify and reproduce robotic motions. Whereas a bottom-up approach is the more promising method in the beginning, generalizing implemented concepts and transferring them to new problems is a challenging task. Unlike this a top-down approach is more difficult to derive in the beginning as specific

application might be missing, but generally easier to adapt as the underlying framework is of generic nature.

The problem associated with all PbD approaches based on HMMs, GMR, DMPs or an explicit optimization [49, 261, 263, 326] require either an explicit learning step or the optimization of a complex nonlinear function and are thus not real-time capable if a movement has to be adapted right after demonstration. On the other hand, movement retargeting approaches specifically designed for online execution [20, 175, 250] show only limited generalization capabilities as the underlying optimization problem is of highly specific structure. It is thus an open question how to combine the universality of learning-based approaches with the real-time execution speed of online motion retargeting approaches.

The contribution of this chapter is an approach based upon one of the fastest known optimization methods - the least-squares optimization. Its solution is calculated by means of the pseudoinverse, which can be given both in closed-form and calculated analytically. The method also provides a way of prioritizing constraints through adequate weights. Differing from linear optimization where conflicting constraints can lead to an empty solution set, LS provides an optimal solution in any case which is advantageous from a practical view point. The adaptation process in this paper is inspired by the discrete Laplace-Beltrami (Laplace) operator which has been used extensively throughout the last decade to deform [7, 6, 38, 219, 279, 281], classify [197, 251] and compress [145, 190, 282] triangular surface meshes in the field of computer vision. As the Laplace operator is extensively studied in literature, it has a well-backed theoretical background [70, 91, 106, 114, 190, 248, 294, 318]. By representing the robot motion by a trajectory, consisting of an n-dimensional path with associated temporal information, the problem is reduced to keep the geometric trajectory properties as similar as possible during path deformation. When interpreting the path as an extremely simple, degenerated surface mesh, the Laplace operator can be applied in a straightforward way to capture the intrinsic path properties. Being a linear operator, the resulting optimization problem is solved through LS. The main problem associated with the approach is to find the right constraints in linear form to achieve additional goals like collision avoidance or a synchronized manipulation involving multiple manipulators.

Simple as it looks, there are also a pitfalls that must be taken care. As such the Laplacian operator is defined only inconsistently across literature. Being mainly based upon the connectivity of the underlying graph, the used Laplace operator belongs to the class of combinatorial mesh Laplacians [324]. Another class are geometric mesh Laplacians, explicitly taking into account the underlying Riemannian geometry [70, 202, 285]. Even if they do capture the geometric properties better, they are only defined on triangle meshes and thus not straightforwardly applicable to paths. The second one is related to the application to trajectories. On the one hand the special mesh structure of paths for simplifications which speed calculations up and help for a better understanding. On the other hand the LS solution suffers from a few drawbacks which are tackled in this chapter, mainly bad scaling effects and the improper handling of nonlinear deformation effects [325].

2.4 Laplacian trajectory editing (LTE)

This section presents LTE as the underlying framework being used throughout this thesis to deform discretized trajectories. A fast and optimal solution is derived that maintains the intrinsic trajectory properties while sufficing additional constraints imposed by the user or the environment. It is assumed that a discretized trajectory is given, consisting of a path described by an ordered set of sampling points $\mathbf{P} = [\mathbf{p}(t_1), \mathbf{p}(t_2), \ldots, \mathbf{p}(t_n)]^T \in \mathbb{R}^{n \times m}$ with associated temporal information $t_i \in \mathbb{R}$ for each sampling point. For simplicity we write $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n]^T$ instead of $\mathbf{P} = [\mathbf{p}(t_1), \mathbf{p}(t_2), \ldots, \mathbf{p}(t_n)]^T$.

During adaptation it is desired to preserve certain properties of the trajectory. The properties are either on a local (e.g. acceleration along the trajectory) or global scale (e.g. absolute position in space). By keeping the temporal information fixed, the problem is reduced to preserve the path properties. To provide a quick solution, the local path properties are described by a set of linear equations

$$\mathbf{LP} = \mathbf{\Delta}, \tag{2.2}$$

where the matrix $\mathbf{L}$ can be interpreted as a operator transforming the path $\mathbf{P}$ into local path properties $\mathbf{\Delta}$.

During adaptation the path has to fulfill additional constraints. Writing them as a linear equation system yields

$$\bar{\mathbf{P}}\mathbf{P} = \bar{\mathbf{C}},$$

with the matrices $\bar{\mathbf{P}}$ and $\bar{\mathbf{C}}$ specifying the type of constraints. There are different ways of solving the two equation systems. If they are linearly independent and of full rank, they can be concatenated and solved for $\mathbf{P}_s = [\mathbf{p}_{1,s}, \mathbf{p}_{2,s}, \ldots, \mathbf{p}_{n,s}]^T \in \mathbb{R}^{n \times m}$ using the matrix inverse as

$$\mathbf{P}_s = \begin{bmatrix} \mathbf{L} \\ \bar{\mathbf{P}} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{\Delta} \\ \bar{\mathbf{C}} \end{bmatrix}.$$

If they are not linearly independent, they have to be prioritized. With the constraints given a higher priority than the local path properties, the problem is rewritten as

$$\begin{aligned} &\min \|\mathbf{LP}_s - \mathbf{\Delta}\|, \\ &\text{s.t. } \bar{\mathbf{P}}\mathbf{P}_s = \bar{\mathbf{C}}, \end{aligned} \tag{2.3}$$

that is a linear optimization problem. An approximate solution of (2.3) can be calculated using LS. By weighting the constraints with a diagonal weighting matrix $\mathbf{\Omega} = \text{diag}(\omega, \ldots, \omega) \gg 1$ the problem is reformulated and solved as

$$\mathbf{P}_s = \begin{bmatrix} \mathbf{L} \\ \mathbf{\Omega}\bar{\mathbf{P}} \end{bmatrix}^{+} \begin{bmatrix} \mathbf{\Delta} \\ \mathbf{\Omega}\bar{\mathbf{C}} \end{bmatrix}. \tag{2.4}$$

Differing from (2.3) where the solution set can be empty, there is always a solution found in (2.4) which is advantageous from a practical point. In addition, if the LS problem is sparse, it has linear computational complexity and can be solved efficiently. The downside is that except for $\mathbf{\Omega} \to \infty$ the constraints are never met perfectly.

Due to the additional boundary constraints in $\bar{\mathbf{P}}$ and $\bar{\mathbf{C}}$, the paths $\mathbf{P}_s$ and $\mathbf{P}$ generally differ from each other. This constitutes the core concept of the trajectory adaptation scheme. The original trajectory $\mathbf{P}$ is only needed in order to define the local path properties $\boldsymbol{\Delta}$ that have to be preserved. After they are determined and the matrices $\mathbf{L}, \bar{\mathbf{P}}, \bar{\mathbf{C}}$ are given, the resulting deformed path $\mathbf{P}_s$ is directly calculated using LS according to (2.4).

Both local path properties as in (2.2) and constraints as in (2.2) are described by the same set of equations. The only difference is that local path properties are based upon a similar type of equation for all sampling points whereas the constraints apply only to a few sampling points. The first and probably most important type are positional constraints of the form

$$\mathbf{p}_i = \mathbf{c}_{i0}, \tag{2.5}$$

pinning a sampling point $\mathbf{p}_i$ to a desired position $\mathbf{c}_{i0}$. For the local path properties this results in the identity matrix as

$$\mathbf{L} = \mathbf{L}_0 = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}. \tag{2.6}$$

A drawback of the matrix $\mathbf{L}_0$ is the decoupling of subsequent sampling point, thus treating every sampling point independently. Another option coupling subsequent sampling points are first order finite difference constraints of the form

$$\mathbf{p}_i - \mathbf{p}_{i-1} = \mathbf{c}_{i1}, \tag{2.7}$$

resulting in a fixed spatial difference between two sampling points $\mathbf{p}_{i+1}$ and $\mathbf{p}_i$ along the path. In matrix form the local path properties are described by

$$\mathbf{L} = \mathbf{L}_1 = \begin{bmatrix} -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & \\ & & & & -1 & 1 \end{bmatrix}. \tag{2.8}$$

The scheme of (2.5) and (2.7) can be extended to higher order finite differences. Hence it is for second order central finite differences

$$\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1} = \mathbf{c}_{i2}, \tag{2.9}$$

resulting in the matrix

$$\mathbf{L} = \mathbf{L}_2 = \begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 & 1 \end{bmatrix}. \tag{2.10}$$

In (2.7)-(2.10) the matrix $\mathbf{L}$ consists only of one specific type of equation. Yet they can also be concatenated and weighted depending on the user preferences. For example a weighted instance of both first and second order finite difference for the local path properties with additional constraints results in

$$\mathbf{L} = \begin{bmatrix} \mathbf{\Omega}_1\mathbf{L}_1 \\ \mathbf{\Omega}_2\mathbf{L}_2 \end{bmatrix}, \qquad \mathbf{\Delta} = \begin{bmatrix} \mathbf{\Omega}_1\mathbf{\Delta}_1 \\ \mathbf{\Omega}_2\mathbf{\Delta}_2 \end{bmatrix}, \tag{2.11}$$

$$\mathbf{P}_s = \begin{bmatrix} \mathbf{L} \\ \mathbf{\Omega}\bar{\mathbf{P}} \end{bmatrix}^+ \begin{bmatrix} \mathbf{\Delta} \\ \mathbf{\Omega}\bar{\mathbf{C}} \end{bmatrix} = \begin{bmatrix} \mathbf{\Omega}_1\mathbf{L}_1 \\ \mathbf{\Omega}_2\mathbf{L}_2 \\ \mathbf{\Omega}\bar{\mathbf{P}} \end{bmatrix}^+ \begin{bmatrix} \mathbf{\Omega}_1\mathbf{\Delta}_1 \\ \mathbf{\Omega}_2\mathbf{\Delta}_2 \\ \mathbf{\Omega}\bar{\mathbf{C}} \end{bmatrix}, \tag{2.12}$$

with $\mathbf{\Omega} \gg \mathbf{\Omega}_1, \mathbf{\Omega}_2$ as three diagonal weighting matrices. If the matrices $\mathbf{L}$ and $\bar{\mathbf{P}}$ consist of arbitrary elements, it must be noted that when writing the first n_z finite differences in matrix form as

$$\begin{bmatrix} 1 & & & & & \\ -1 & 1 & & & \mathbf{0} & \\ 1 & -2 & 1 & & & \\ -1 & 3 & -3 & 1 & & \\ 1 & -4 & 6 & -4 & 1 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots, \end{bmatrix},$$

one can see that they are linearly independent. Hence any row of $\mathbf{L}$ or $\bar{\mathbf{P}}$ with n_z nonzero entries can be interpreted as a weighted sum of the first n_z order derivatives.

Fig. 2.1 shows a small deformation example using LTE under the influence of positional constraints to display its capabilities. Whereas the upper row depicts the unmodified path $\mathbf{P}$, the lower row shows four different realizations where specific sampling points (round dots) are pinned to defined positions.

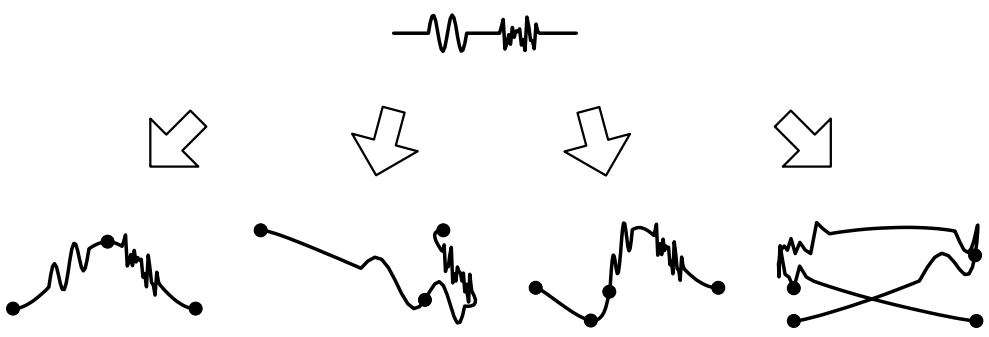


Figure 2.1: Various possible path deformations using LTE by applying positional constraints to individual sampling points (round dots)

Laplacian trajectory editing

Given a path $\mathbf{P}$ described by its local path properties

$$\mathbf{LP} = \boldsymbol{\Delta},$$

additonal constraints are imposed in linear form as

$$\bar{\mathbf{P}}\mathbf{P} = \bar{\mathbf{C}}.$$

Then an optimal solution of the deformed path $\mathbf{P}_s$ is calculated using least squares as

$$\mathbf{P}_s = \begin{bmatrix} \mathbf{L} \\ \boldsymbol{\Omega}\bar{\mathbf{P}} \end{bmatrix}^{+} \begin{bmatrix} \boldsymbol{\Delta} \\ \boldsymbol{\Omega}\bar{\mathbf{C}} \end{bmatrix},$$

with the diagonal weighting matrix $\boldsymbol{\Omega}$.

2.5 Properties of LTE

This section provides a detailed analysis of the theoretical properties of LTE and how they can be applied to account for a wider class of problems. In addition, it examines the relation of LTE to other disciplines like computer graphics.

Derivation from Laplacian mesh editing

Before applied to trajectories, the Laplace-Beltrami operator has been used in computer graphic for more than one decade to deform surface meshes [278]. This method is known as Laplacian mesh editing. When deriving LTE from Laplacian mesh editing, the basic principle is to interpret a discretized path as an extremely primitive, 1D surface mesh and apply the same geometric mesh deformation operations to it. Whereas some results are identical, others derived from two-dimensional (2D) Riemannian manifolds do not have any equivalent when applied to paths. When interpreting the path like a surface mesh as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each vertex v_i is associated with one sampling point $\mathbf{p}_i$, the neighbor set $\mathcal{N}_i$ of the vertex v_i is the set of all adjacent vertices v_j and the edge set is defined as $\mathcal{E} = \{e_{ij}\}, i, j \in \{1, .., n\}$ with

$$e_{ij} = \begin{cases} w_{ij} & \text{if } j \in \mathcal{N}_i, \\ 0 & \text{otherwise.} \end{cases}$$

Instead of working in absolute Cartesian coordinates, the discrete Laplace-Beltrami operator $D(v_i)$ or Laplacian is used to specify the local path properties, called Laplacian coordinates $\boldsymbol{\delta}_{g,i}$ [194]. For vertex v_i, this results in

$$D(v_i) = \boldsymbol{\delta}_{g,i} = \sum_{j \in \mathcal{N}_i} \frac{w_{ij}}{\sum\limits_{j \in \mathcal{N}_i} w_{ij}} (\mathbf{p}_i - \mathbf{p}_j).$$

Applied to surface meshes, the Laplacian coordinates specify the mean curvature of the mesh at each vertex v_i. When used for paths, they serve as an approximation of the curvature along the path. Written in matrix form, the discrete Laplace-Beltrami operator resembles the graph Laplacian matrix $\mathbf{L}_g \in \mathbb{R}^{n \times n}$ encoding the topology of the graph as

$$\mathbf{L}_{g\{ij\}} = \begin{cases} 1 & \text{if } i = j, \\ -\dfrac{w_{ij}}{\sum\limits_{j \in \mathcal{N}_i} w_{ij}} & \text{if } j \in \mathcal{N}_i, \\ 0 & \text{otherwise.} \end{cases}$$

By defining $w_{ij} = 1$, one obtains the typical structure for paths as

$$\mathbf{L}_g = \frac{1}{2} \begin{bmatrix} 2 & -2 & & & & & & \\ -1 & 2 & -1 & & & & & \\ & -1 & 2 & -1 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & -1 & 2 & -1 & & \\ & & & & -1 & 2 & -1 & \\ & & & & & -2 & 2 & \end{bmatrix}.$$

When concatenating all Laplacian coordinates $\boldsymbol{\delta}_{g,i}$ into a single matrix $\boldsymbol{\Delta}_g = \left[\boldsymbol{\delta}_{g,1}, \boldsymbol{\delta}_{g,2} \ldots, \boldsymbol{\delta}_{g,n}\right]^T$, it becomes

$$\mathbf{L}_g \mathbf{P} = \boldsymbol{\Delta}_g,$$

which is equivalent to (2.2) with $\mathbf{L} = \mathbf{L}_g$. In this case the matrix $\mathbf{L}$ consists of first order finite differences for the first and last sampling point and second order finite differences for all other sampling points of the path.

It shall be noted that neither the Laplacian operator nor the edge weights w_{ij} are uniquely defined in the field of Laplacian mesh editing. Starting with the Laplacian operator, there are two cases that have to be distinguished [324]: Combinatorial mesh Laplacians and geometric mesh Laplacians. Combinatorial mesh Laplacians [57, 67, 293, 323] are based entirely upon topological information of the underlying graph, i.e. the edge connectivity. Then the geometry of the underlying mesh structure is only defined indirectly, requiring a "well-defined" graph structure with similar edge length. Geometric mesh Laplacians [202, 246] on the other hand are directly derived by discretizing the continuous Laplace-Beltrami operator on a Riemannian manifold [118, 210]. They implicitly considers the manifold geometry, yet are only defined on triangle meshes and thus not applicable for trajectories. To account for irregularly shaped meshes, different edge weight schemes have been developed. Among the most simple ones are uniform umbrella weights, resulting in $w_{ij} = 1$. In this case it is assumed that the edge length $l_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|_2$ is roughly equal for all graph edges. In case of irregular lengths, a better choice are scale-dependent umbrella weights $w_{ij} = 1/l_{ij}$. Another option are cotangent weights [67, 246], yet they are only defined on triangle meshes and thus not applicable to paths. Last, one can also directly modify the position of all sampling points through smoothing [67, 223] causing an irreversible effect.

Interpretation

Despite the mathematical background originating from Riemannian manifolds, there are intuitive geometric and physical interpretations of LTE available.

To derive an interpretation for the discretized path, we look at the continuous path $\mathbf{\Pi}(s) \in \mathbb{R}^3$ parameterized by the arc length s. Then the Frenet-Serret formula describe the local curvature κ and the local torsion τ. By introducing the unit tangent vector $\mathbf{t}$, the normal unit vector $\mathbf{n}$ and the binormal unit vector $\mathbf{b} = \mathbf{t} \times \mathbf{n}$ with $\times$ as the three-dimensional (3D) cross product, one obtains [73]

$$\begin{aligned} \frac{d\mathbf{t}}{ds} &= \kappa\mathbf{n}, \\ \frac{d\mathbf{n}}{ds} &= -\kappa\mathbf{t} + \tau\mathbf{b}, \\ \frac{d\mathbf{b}}{ds} &= -\tau\mathbf{n}. \end{aligned} \tag{2.13}$$

The curvature κ is calculated as

$$\kappa = \left\| \frac{d\mathbf{t}}{ds} \right\| = \left\| \frac{d^2\mathbf{\Pi}(s)}{ds^2} \right\|. \tag{2.14}$$

In case of a discrete path $\mathbf{P}$ and $m = 3$ the central difference approximation of (2.14) is

$$\left\| \frac{d^2\mathbf{p}(s)}{d\mathbf{s}^2} \right\| = \frac{\mathbf{p}(s+h) - 2\mathbf{p}(s) + \mathbf{p}(s-h)}{h^2}, \tag{2.15}$$

with step length h. If the sampling points are equidistantly spaced with distance h, (2.15) is expressed as

$$\left\| \frac{d^2\boldsymbol{p}_i}{ds^2} \right\| = \frac{\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1}}{h^2},$$

This formula is closely related to the one for Laplacian coordinates, formulated in [293] as

$$\boldsymbol{\delta}_i = \frac{\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1}}{-2}, \tag{2.16}$$

with the scaling factor $\frac{1}{h^2}$ instead of $\frac{1}{-2}$ as the only difference. If the trajectory sampling points are also sampled at a constant sampling rate, it results in a constant temporal difference $\Delta t = t_i - t_{i-1}$ between subsequent sampling points. Then the acceleration along the trajectory is expressed as

$$\ddot{\mathbf{p}}_i = \frac{\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1}}{\Delta t^2},$$

differing from (2.16) only by the scaling factor $\frac{1}{\Delta t^2}$ instead of $\frac{1}{-2}$. If two subsequent sampling points are fixed as described by

$$\dot{\mathbf{p}}_i = \frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{\Delta t},$$

it corresponds to a velocity constraints. In a similar fashion three subsequent sampling points correspond to acceleration, four to jerk, and so on.

Handling nonlinear deformation effects

Based upon a set of linear equations, LTE cannot handle nonlinear deformation effects like rotations. Whereas nonlinear deformation effects are negligible for small deformations, they play a major role as the amount of deformation increases [196], see Fig. 2.2. Shown is a handwritten word ("Hello") which is deformed by fixing three sampling points using positional constraints. The left side shows the result when ignoring nonlinear deformation effects during path adaptation. In contrast a multiresolution approach described later in this chapter explicitly considers nonlinear deformation effects and resembles the original word well as shown on the right side. To handle nonlinear deformation effects on

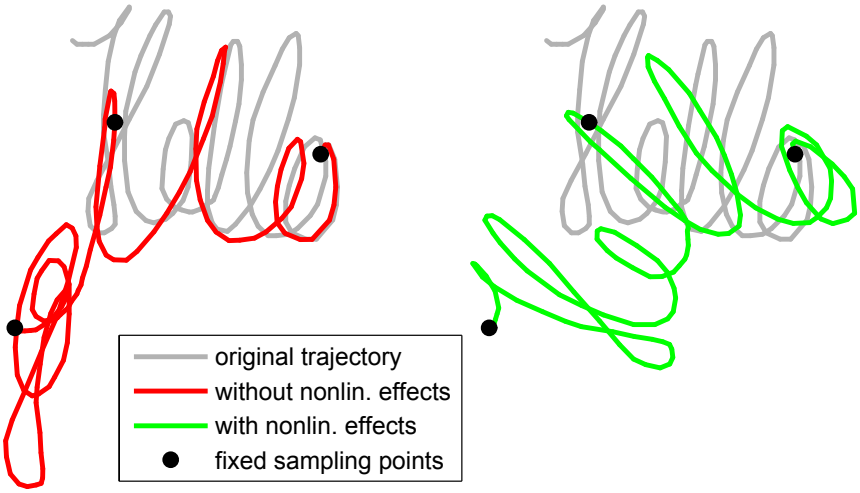


Figure 2.2: Influence of nonlinear deformation effects during path adaptation

a local path scale, the method of Arun [16, 304] is combined with LTE. Similar to [124, 195] it finds an optimal nonlinear deformation for each vertex such that the amount of intrinsic distortion is minimized.

The core concept of the method of Arun is recapitulated as follows: Assume that one is given two sets of sampling points, namely $\mathbf{P}_r = [\mathbf{p}_{r,1}, \mathbf{p}_{r,2}, \ldots, \mathbf{p}_{r,k}]^T \in \mathbb{R}^{k\times m}$ and $\mathbf{P}_d = [\mathbf{p}_{d,1}, \mathbf{p}_{d,2}, \ldots, \mathbf{p}_{d,k}]^T \in \mathbb{R}^{k\times m}$ that have to be matched using the homogeneous transformation

$$\mathbf{p}_{d,i} = \hat{c}\hat{\mathbf{R}}\mathbf{p}_{r,i} + \hat{\mathbf{t}} \qquad \forall i = 1, \ldots, k,$$

with constant $\hat{c}$ as a scalar scaling factor, $\hat{\mathbf{R}} \in \mathbb{R}^{m\times m}$ as a rotation matrix and $\hat{\mathbf{t}} \in \mathbb{R}^m$ as a translational vector. Due to matching imperfections we are looking for the best possible transformation between $\mathbf{P}_r$ and $\mathbf{P}_d$, resulting in a minimization of the error term $\mathbf{n}_o$ as

$$\mathbf{n}_o = \sum_{i=1}^{k} \left\| \mathbf{p}_{d,i} - (\hat{c}\hat{\mathbf{R}}\mathbf{p}_{r,i} + \hat{\mathbf{t}}) \right\|^2. \tag{2.17}$$

The solution is calculated through a singular value decomposition (SVD): Let $\mathbf{Q} \in \mathbb{R}^{m\times m}$ be the covariance matrix as

$$\mathbf{Q} = \frac{1}{k}\sum_{i=1}^{k} (\mathbf{p}_{ri} - \bar{\mathbf{p}}_r)(\mathbf{p}_{d,i} - \bar{\mathbf{p}}_d)^T,$$

with $\bar{\mathbf{p}}_r$ as the centroid of $\mathbf{P}_r$ and $\bar{\mathbf{p}}_d$ as the centroid of $\mathbf{P}_d$

$$\bar{\mathbf{p}}_r = \frac{1}{k}\sum_{i=1}^{k}\mathbf{p}_{r,i}, \qquad \bar{\mathbf{p}}_d = \frac{1}{k}\sum_{i=1}^{k}\mathbf{p}_{d,i}.$$

Similarly, the variance σ_s^2 is calculated as

$$\sigma_s^2 = \frac{1}{k}\sum_{i=1}^{k}\left\|\mathbf{p}'_{r,i}\right\|^2.$$

The SVD of $\mathbf{Q}$ is calculated such that

$$\mathbf{Q} = \mathbf{U}\mathbf{S}\mathbf{V}^T.$$

Then $\hat{c}$, $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ are computed as

$$\begin{aligned}\hat{\mathbf{R}} &= \mathbf{V}\mathbf{S}'\mathbf{U}^T,\\ \hat{c} &= \frac{1}{\sigma_s^2}\operatorname{tr}(\mathbf{S}\mathbf{S}'),\\ \hat{\mathbf{t}} &= \bar{\mathbf{p}}_d - \hat{c}\hat{\mathbf{R}}\bar{\mathbf{p}}_s.\end{aligned}$$

with $\mathbf{S}'$ preventing mirrored mappings

$$\mathbf{S}' = \begin{cases}\mathbf{I} & \text{if } \det(U)\det(V) = 1,\\ \operatorname{diag}(1,\ldots,1,-1) & \text{if } \det(U)\det(V) = -1.\end{cases}$$

As shown in [280], the method can be adapted with $c = 1$ to rotate the Laplacian coordinates $\boldsymbol{\delta}_i$ of a mesh individually for every sampling point, named as-rigid-as-possible (ARAP) deformation. This results in new Laplacian coordinates $\hat{\boldsymbol{\delta}}_i$ as

$$\hat{\boldsymbol{\delta}}_i = \hat{\mathbf{R}}_i\boldsymbol{\delta}_i,$$

with the rotation matrix $\hat{\mathbf{R}}_i$ based upon the sampling points' position of original and deformed mesh. The method can also be applied to paths. For paths, the two sets of sampling points are $\mathbf{P}_r = [\mathbf{p}_i - \mathbf{p}_{i-1}, \mathbf{p}_i - \mathbf{p}_{i+1}]^T$ and $\mathbf{P}_d = [\mathbf{p}_{i,s} - \mathbf{p}_{i-1,s}, \mathbf{p}_{i,s} - \mathbf{p}_{i+1,s}]^T$. We realize that both $\mathbf{P}_r$ and $\mathbf{P}_d$ consist only of two vectors. Then an optimal rotation $\mathbf{R}_i$ is calculated in 2D and 3D using basic geometry, making the use of a SVD obsolete.

Path similarity measure

LTE can be used to measure the similarity between two given paths. Depending on whether nonlinear deformation effects are taken into account, the resulting measure is either intrinsically provided by the residual of the LS solution or by the underlying cost function which is minimized when handling nonlinear deformation effects. When neglecting nonlinear deformation effects, the LS solution in (2.4) minimizes the cost term E defined as

$$E = \left\|\begin{bmatrix}\mathbf{L}\\ \boldsymbol{\Omega}\bar{\mathbf{P}}\end{bmatrix}\mathbf{P}_s - \begin{bmatrix}\boldsymbol{\Delta}\\ \boldsymbol{\Omega}\bar{\mathbf{C}}\end{bmatrix}\right\|_F^2,$$

with the subscript $_F$ as the Frobenius norm. If positional constraints are the only type of constraints, the unweighted positional error $\|\bar{\mathbf{P}} - \bar{\mathbf{C}}\| \approx 0$ is invisible to the human eye. In this case a more human-oriented measure E_1 is given as

$$E_1 = \left\| \mathbf{L}\mathbf{P}_s - \boldsymbol{\Delta} \right\|_F^2 . \tag{2.18}$$

If nonlinear deformation effects shall be taken into account as well, [280] showed that an optimal solution of the weighted instance of (2.17) corresponds to minimizing the cost term E_2, defined as

$$E_2 = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} w_{ij} \| (\mathbf{p}_j - \mathbf{p}_i) - \hat{\mathbf{R}}_i (\mathbf{p}_{j,s} - \mathbf{p}_{i,s}) \|_2^2 .$$

with the rotational matrix $\hat{\mathbf{R}}_i$ as calculated in Sec. 2.5 such that the deformed sampling point positions $\mathbf{p}_{j,s}, \mathbf{p}_{i,s} \in \mathbf{P}_s$ match the original sampling points $\mathbf{p}_j, \mathbf{p}_i \in \mathbf{P}$ best. If the measure E_3 should not only account for rotational but also scaling effects, it has to be modified. This gives the new measure E_3 as

$$E_3 = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} w_{ij} \| (\mathbf{p}_j - \mathbf{p}_i) - c\hat{\mathbf{R}}_i (\mathbf{p}_{j,s} - \mathbf{p}_{i,s}) \|_2^2 .$$

with the scale factor c as in (2.17) as the only difference. A last commonly used measure to calculate the similarity between two paths is the summed quadratic difference of the original respectively deformed path's absolute sampling points' positions. This gives the measure E_4 as

$$E_4 = \sum_{i=1}^{n} \| \mathbf{p}_{i,s} - \mathbf{p}_i \|_2^2 , \tag{2.19}$$

for the sampling point position $\mathbf{p}_i$ of the original path and the sampling point position $\mathbf{p}_{i,s}$ of the deformed path.

As it is unknown how LTE is perceived by humans, a psychological experiment is performed to resolve this issue. Five synthetic 3D reference paths composed of added sinusoidal paths and five real 3D reference path recorded with a Kinect are deformed randomly using the multiresolution approach presented in Sec. 2.7. Based upon every reference path three deformed paths are calculated, see Fig. 2.3. Ten participants (male/22-32yrs/∅25yrs) are asked to rate "how similar" each path is with respect to the corresponding reference path on a scale from 0 (very similar) to 9 (very dissimilar). The result (error bars with mean and standard deviation) for all four measures E_1-E_4 together with the Pearson correlation coefficient r for a semilogarithmic plot are given in Fig. 2.3, showing a good correspondence between the logarithmic value of the similarity measure and the perceived human similarity. Apparently the measure E_2 reflects the human perception of path similarity best, the measure E_4 least. A detailed description of the experiment is given in App. A.

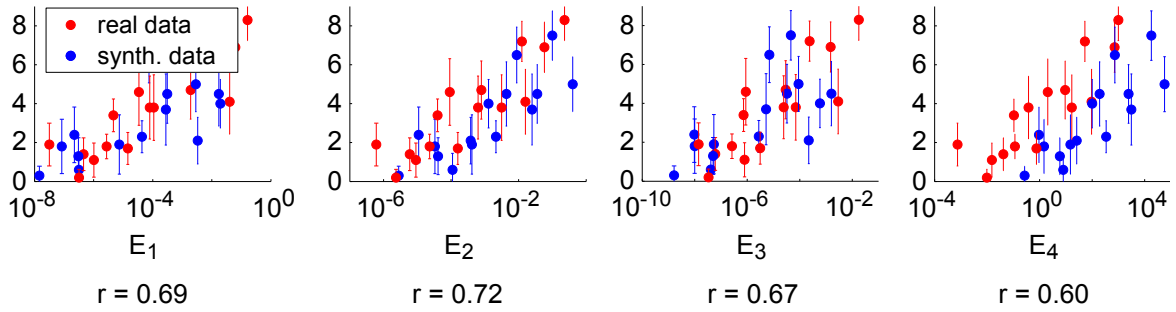


Figure 2.3: Evaluation of the psychological experiment for the four similarity measures E_1-E_4 with corresponding Pearson correlation coefficient r

2.6 Spatial bounds on the LTE deformation

Calculating the pseudoinverse in (2.4) is computationally demanding for trajectories with many sampling points. As the exact shape of the resulting trajectory after the LTE deformation is unknown, multiple deformations are necessary to fit the desired path to external constraints like workspace boundaries or obstacles along the path. To reduce the computational load, this section aims at determining spatial bounds of the LTE solution that can be calculated without the need to evaluate the LS solution. Two different methods are presented. The first method interprets the path deformation process as the coordinated movement of a leader-follower network, which allows to decomposed the LS solution into a set of smaller problems which are solved independently and provide tighter spatial bounds as they do not suffer from numerical problems. The second method finds an equivalent spline representation for LTE in the continuous domain. By representing splines through Bernstein polynomials, tight spatial bounds are derived.

2.6.1 Spatial bounds based on multi-agent systems

This section derives a tight upper bound based on the norm of the LTE solution that is not affected by numerical instabilities. A standard way to calculate such a bound is based on the Cauchy-Schwarz inequality as

$$\|\mathbf{P}_s\| \leq \left\| \begin{bmatrix} \mathbf{L} \\ \mathbf{\Omega}\bar{\mathbf{P}} \end{bmatrix}^{+} \right\| \left\| \begin{bmatrix} \mathbf{\Delta} \\ \mathbf{\Omega}\bar{\mathbf{C}} \end{bmatrix} \right\|. \tag{2.20}$$

Yet this approach suffers from numerical problems as the matrix $[\mathbf{L}^T \;\; \bar{\mathbf{P}}^T\mathbf{\Omega}^T]$ is badly conditioned due to large weights of the weighting matrix $\mathbf{\Omega}$. Rather than preconditioning the matrix [27], the matrix $[\mathbf{L}^T \;\; \bar{\mathbf{P}}^T\mathbf{\Omega}^T]$ is split up into several submatrices. For the limiting case $\|\mathbf{\Omega}\| \to \infty$ the LS solution then becomes independent of the weighting matrix $\mathbf{\Omega}$. A gradient descent approach is used to derive a solution. If the additional constraints consist only of positional constraints, the movement of all sampling points during gradient descent is described by a dynamical system of interconnected agents.

The LS solution in (2.4) minimizes the cost term E as

$$\mathbf{P}_s = \underset{\mathbf{P}}{\operatorname{argmin}}(E) = \underset{\mathbf{P}}{\operatorname{argmin}} \left\| \begin{bmatrix} \mathbf{L} \\ \mathbf{\Omega}\bar{\mathbf{P}} \end{bmatrix} \mathbf{P} - \begin{bmatrix} \mathbf{\Delta} \\ \mathbf{\Omega}\bar{\mathbf{C}} \end{bmatrix} \right\|_F^2 .$$

Calculating the partial derivative $\frac{\partial E}{\partial \mathbf{P}}$ independently along every dimension (without writing the subscript ${}_{\{:k\}}$) results in

$$\frac{\partial E}{\partial \mathbf{P}} = 2(\mathbf{L}^T\mathbf{L} + \bar{\mathbf{P}}^T\mathbf{\Omega}^2\bar{\mathbf{P}})\mathbf{P} - 2\mathbf{L}^T\mathbf{\Delta} - 2\bar{\mathbf{P}}^T\mathbf{\Omega}^2\mathbf{C}.$$

Note that $\mathbf{\Omega}^T = \mathbf{\Omega}$ because it is a diagonal weighting matrix. A gradient descent approach is used to find an optimal solution. When descending along the steepest gradient, it becomes

$$\dot{\mathbf{P}} = -(\mathbf{L}^T\mathbf{L} + \bar{\mathbf{P}}^T\mathbf{\Omega}^2\bar{\mathbf{P}})\mathbf{P} + \mathbf{L}^T\mathbf{\Delta} + \bar{\mathbf{P}}^T\mathbf{\Omega}^2\mathbf{C}, \tag{2.21}$$

which is interpreted as a dynamical system of the form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{u},$$

with

$$\begin{aligned} \mathbf{x} &= \mathbf{P}, \\ \dot{\mathbf{x}} &= \dot{\mathbf{P}}, \\ \mathbf{A} &= -\mathbf{L}^T\mathbf{L} - \bar{\mathbf{P}}^T\mathbf{\Omega}^2\bar{\mathbf{P}}, \\ \mathbf{u} &= \mathbf{L}^T\mathbf{\Delta} + \bar{\mathbf{P}}^T\mathbf{\Omega}^2\mathbf{C}. \end{aligned}$$

If the matrix $[\mathbf{L}^T \;\; \bar{\mathbf{P}}^T\mathbf{\Omega}^T]^T$ has full column rank n, the position of all sampling points in $\mathbf{P}_s$ are uniquely specified by the LS solution. In this case the resulting dynamical system is stable as it performs a gradient descent towards the optimal solution of a convex function. Inspired by multi agent systems, (2.21) is split up into sampling points with fixed positional constraints denoted by a subscript ${}_L$ and all other sampling points with subscript ${}_F$. This keeps the notation of leader sampling points with subscript ${}_L$ and follower sampling points with subscript ${}_F$ consistent with literature [78]. Rearranging the elements in $\mathbf{P}$ results in

$$\begin{bmatrix} \dot{\mathbf{P}}_L \\ \dot{\mathbf{P}}_F \end{bmatrix} = -\begin{bmatrix} (\mathbf{L}^T\mathbf{L})_L + \bar{\mathbf{\Omega}}^2 & (\mathbf{L}^T\mathbf{L})_{LF} \\ (\mathbf{L}^T\mathbf{L})_{LF}^T & (\mathbf{L}^T\mathbf{L})_F \end{bmatrix} \begin{bmatrix} \mathbf{P}_L \\ \mathbf{P}_F \end{bmatrix} + \begin{bmatrix} \mathbf{L}_L^T\mathbf{\Delta}_L + \mathbf{\Omega}^2\mathbf{C}' \\ \mathbf{L}_F^T\mathbf{\Delta}_F \end{bmatrix}, \tag{2.22}$$

with the matrix $\mathbf{L}^T\mathbf{L}$ being split up into three submatrices $(\mathbf{L}^T\mathbf{L})_L$, $(\mathbf{L}^T\mathbf{L})_F$ and $(\mathbf{L}^T\mathbf{L})_{LF}$ accounting for the autonomous leader movement, the autonomous follower movement and the follower movement induced by the leader movement. In a similar fashion, the matrix $\mathbf{\Delta}$ is split up into two parts $\mathbf{\Delta}_L$ and $\mathbf{\Delta}_F$. The matrix $\bar{\mathbf{P}}^T\mathbf{\Omega}^2\bar{\mathbf{P}}$ is reduced to $\mathbf{\Omega}^2$ in this representation and the matrix $\bar{\mathbf{P}}^T\mathbf{\Omega}^2\mathbf{C}$ is expressed by $\mathbf{\Omega}^2\mathbf{C}'$ with the matrix $\mathbf{C}'$ as a row-permuted version of $\mathbf{C}$. When prioritizing the positional constraints as $\|\mathbf{\Omega}\| \to \infty$, (2.22) reduces to

$$\begin{bmatrix} \dot{\mathbf{P}}_L \\ \dot{\mathbf{P}}_F \end{bmatrix} = \begin{bmatrix} -\mathbf{\Omega}^2\mathbf{P}_L + \mathbf{\Omega}^2\mathbf{C}' \\ -(\mathbf{L}^T\mathbf{L})_{LF}^T\mathbf{P}_L - (\mathbf{L}^T\mathbf{L})_F\mathbf{P}_F + \mathbf{L}_F^T\mathbf{\Delta}_F \end{bmatrix}. \tag{2.23}$$

Setting the left side of (2.23) to zero results in

$$\begin{aligned}\mathbf{P}_L =& \mathbf{C},\\ \mathbf{P}_F =& -(\mathbf{L}^T\mathbf{L})_F^{-1}\left((\mathbf{L}^T\mathbf{L})_{LF}^T\mathbf{C}-\mathbf{L}_F^T\boldsymbol{\Delta}_F\right),\end{aligned}$$

proving that the leader sampling points converge to their desired position $\mathbf{C}$. Differing from (2.21) the solution in (2.23) is now independent of the weighting factors $\boldsymbol{\Omega}$. By splitting the term $\mathbf{C}$ up into a constant factor $\mathbf{C}'$ coinciding with the original path $\mathbf{P}$ and a variable factor $\Delta\mathbf{C}$ accounting for the deformation $\Delta\mathbf{P}$, $\mathbf{P}_F$ is split up in a similar manner into a constant part $\mathbf{P}'_F$ and a variable part $\Delta\mathbf{P}_F$ as

$$\begin{aligned}\mathbf{P}_F &= \mathbf{P}'_F + \Delta\mathbf{P}_F\\ &= -(\mathbf{L}^T\mathbf{L})_F^{-1}\left((\mathbf{L}^T\mathbf{L})_{LF}^T\mathbf{C}'-\mathbf{L}_F^T\boldsymbol{\Delta}\right)\\ &\quad -(\mathbf{L}^T\mathbf{L})_F^{-1}(\mathbf{L}^T\mathbf{L})_{LF}^T\Delta\mathbf{C},\end{aligned}$$

Because the term $\mathbf{P}'_F$ coincides with the undeformed path $\mathbf{P}$, we are only interested in $\Delta\mathbf{P}_F$. Its norm is bounded by

$$\|\Delta\mathbf{P}_F\| \leq \|(\mathbf{L}^T\mathbf{L})_F^{-1}\|\|(\mathbf{L}^T\mathbf{L})_{LF}^T\|\|\Delta\mathbf{C}\|,$$

and in general smaller than the solution of (2.20), resulting in tighter spatial bounds.

2.6.2 Spatial bounds based on spline representation

It is possible for some special cases to obtain a LTE solution without the need to calculate a pseudoinverse by deriving an equivalent solution in the continuous domain. The resulting solution is a set of piecewise polynomial functions, better known as splines. The advantage is a reduced computational complexity as an equation system of much smaller size has to be solved. In addition this solution provides the tightest spatial bounds known for LTE. The pseudoinverse $\left[\mathbf{L}^T(\boldsymbol{\Omega}\bar{\mathbf{P}})^T\right]^{T+}$ is a linear operator. One can thus decompose the solution in (2.4) as

$$\begin{aligned}\mathbf{P}_s =& \underbrace{\begin{bmatrix}\mathbf{L}\\ \boldsymbol{\Omega}\bar{\mathbf{P}}\end{bmatrix}^+\begin{bmatrix}\boldsymbol{\Delta}\\ \boldsymbol{\Omega}\bar{\mathbf{C}}_1\end{bmatrix}}_{\{1\}} + \underbrace{\begin{bmatrix}\mathbf{L}\\ \boldsymbol{\Omega}\bar{\mathbf{P}}\end{bmatrix}^+\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\Omega}\bar{\mathbf{C}}_2\end{bmatrix}}_{\{2\}},\\ \text{s.t. } &\bar{\mathbf{C}}_1+\bar{\mathbf{C}}_2=\bar{\mathbf{C}}\end{aligned}$$

Through a suitable choice of $\bar{\mathbf{C}}_1$, the term $\{1\}$ is a sole translation of the original trajectory $\mathbf{P}$ that can be calculated without the need of an explicit LS solution. Under specific conditions, the term $\{2\}$ can be approximated using splines.

Theorem 1 Let $\mathbf{L} = \mathbf{L}_1, \mathbf{L}_2, \ldots$ be a Laplacian matrix of same order finite differences for all sampling points, $\boldsymbol{\Delta} = \mathbf{0}$ and the matrix $\bar{\mathbf{P}}$ consist only of positional constraints according to (2.5). If furthermore all sampling points of the trajectory are equitemporally spaced as $t_i - t_{i-1} = \Delta t \;\; \forall i \in \{2,\ldots,n\}$, LTE as calculated in (2.4) is identical for $\boldsymbol{\Omega} \to \infty$ to a spline-based approach in the continuous domain.

Proof. Without loss of generality only the most common case $\mathbf{L} = \mathbf{L}_2$ is considered. The optimized trajectory $\mathbf{P}_s$ is then calculated as

$$\mathbf{P}_s = \begin{bmatrix} \mathbf{L} \\ \mathbf{\Omega}\bar{\mathbf{P}} \end{bmatrix}^+ \begin{bmatrix} \mathbf{0} \\ \mathbf{\Omega}\bar{\mathbf{C}} \end{bmatrix}.$$

The underlying problem is to find a trajectory with minimal acceleration going through a set of predefined via points. In the continuous domain this corresponds to minimizing the acceleration $\ddot{x}$. An optimal trajectory x_{opt} is calculated as

$$x_{opt} = \min_x I(x) = \min_x \frac{1}{2} \int_0^T \ddot{x}(t)^2 dt.$$

The derivation of a solution is conceptually similar to the one for minimum-jerk trajectories [90]. Through the calculus of variation we consider the disturbed trajectory $x(t)+\epsilon\eta$ with the scalar ϵ and η as an arbitrary function fulfilling the boundary conditions

$$\begin{aligned} \eta(0) = 0, \quad \eta(T) = 0, \\ \dot{\eta}(0) = 0, \quad \dot{\eta}(T) = 0, \end{aligned} \tag{2.24}$$

This results in

$$I(x+\epsilon\eta) = \frac{1}{2} \int_0^T (\ddot{x} + \epsilon\ddot{\eta})^2 dt,$$

$$\frac{dI(x+\epsilon\eta)}{d\epsilon} = \int_0^T (\ddot{x} + \epsilon\ddot{\eta})\ddot{\eta} dt.$$

For x to minimize $I(x+\epsilon\eta)$, the following condition has to be fulfilled. In all other cases the trajectory x is not optimal. Note that we write $^{(i)}$ for the i-th time derivative.

$$\left.\frac{dI(x+\epsilon\eta)}{d\epsilon}\right|_{\epsilon=0} = 0 = \int_0^T \ddot{x}\ddot{\eta} = x^{(2)}\eta^{(2)}.$$

Through partial integration we obtain

$$\int_0^T x^{(2)}\eta^{(2)} = \underbrace{x^{(2)}\eta^{(1)}\big|_0^T}_{=0} - \int_0^T x^{(3)}\eta^{(1)} = -\int_0^T x^{(3)}\eta^{(1)}.$$

Applying partial integration another time yields

$$-\int_0^T x^{(3)}\eta^{(1)} = -\underbrace{x^{(3)}\eta\big|_0^T}_{=0} + \int_0^T x^{(4)}\eta = \int_0^T x^{(4)}\eta,$$

giving

$$\int_0^T x^{(4)}\eta = 0,$$

as the resulting condition that must hold for any function η. This is the case for any function

$$x^{(4)} = 0 \;\; \forall t \in [0, T].$$

Any third order polynomial of the form

$$x = \sum_{k=0}^{3} \mathbf{a}_k t^k, \tag{2.25}$$

fulfills this condition, resulting in a spline curve if multiple via points must be passed.

■

Differing from LTE, the equation system that has to be solved is independent of the number of sampling points. After the free parameters $\mathbf{a}_k$ are determined, the solution for the discretized trajectory is computed based on the solution (2.25) for the continuous trajectory. However, as there are only four free parameters $a_0, \ldots, a_3$, the number of applicable constraints is limited. For multiple via points that have to be passed, cubic splines with boundary conditions $\ddot{x}(0) = 0$, $\ddot{x}(T) = 0$ are the equivalent continuous representation of $\bar{\mathbf{P}}$ consisting only of positional constraints. For the j-th trajectory interval in a possibly multidimensional space, the resulting cubic spline is written as

$$x_j = \sum_{k=0}^{3} \mathbf{a}_{j,k} t^k, \quad t \in [0, 1], \tag{2.26}$$

There are multiple ways to represent a cubic spline curve. One option are Bézier splines. In this case one can rewrite (2.26) with as

$$x_j = \sum_{k=0}^{l} \mathbf{a}_{j,k} \mathbf{B}_k^l,$$

with the control points $\mathbf{a}_{j,k}$, the parameter $l = 3$ and $\mathbf{B}_k^l$ as Bernstein polynomials of the form

$$\mathbf{B}_k^l(t) = \binom{l}{k} (1-t)^{l-k}\, t^k, \quad t \in [0, 1]. \tag{2.27}$$

Due to the non-negativity of each Bernstein polynomial for $t \in [0, 1]$, every point of $\mathbf{x}_j$ is a convex combination of the control points $\mathbf{a}_{j,k}$. As a result the trajectory segment $\mathbf{x}_j$ always stays within the convex hull of its control points. The same accounts for the discrete trajectory segment $\mathbf{P}_j \in \{2\}$.

The convex hulls conv($\{1\}$) and conv($\{2\}$) are found both for $\{1\}$ through numerical optimization and for $\{2\}$ by calculating the control points of the Bézier curves. The convex hull of the deformed trajectory $\{1\} + \{2\}$ is then bounded from above by

$$\text{conv}(\{1\} + \{2\}) \leq \text{conv}(\{1\} \bigoplus \{2\}) = \text{conv}(\{1\}) \bigoplus \text{conv}(\{2\}), \tag{2.28}$$

with the operator $\bigoplus$ as the Minkowski sum of two convex hulls. Tighter bounds can be derived if the original trajectory is allowed to be split up into smaller trajectory segments for which new convex hulls are calculated independently. Jensen's inequality proves that if a trajectory interval x_j is split up into two smaller trajectory segments $x_{j,1}$ and $x_{j,2}$, it is for the resulting convex hull

$$\text{conv}(x_j) \geq \text{conv}(x_{j,1}) + \text{conv}(x_{j,2}).$$

The presented approach is based on third order polynomials, corresponding to minimum acceleration deformations. The same method can be derived for polynomials of any order, as such fifth order polynomials correspond to minimum jerk deformation, seventh order polynomials to minimum snap deformation, etc.

2.7 Multiresolution LTE for improved performance

The batch LTE method in Sec. 2.4 deforms the entire path by means of a single LS solution. Although being straightforward to apply, the approach is computationally demanding if nonlinear deformation effects have be considered individually for every sampling point along the trajectory. A faster method is to perform all computationally demanding operations only on a small subset of sampling points and interpolate the missing information for all other sampling points afterwards. The method grabs up a similar idea as presented in [77, 111, 158]. The complete approach consists of three steps. In a first downsampling step the number of path sampling points is reduced, resulting in a subset of so called support sampling points. The second step consists of deforming the support sampling points. In the last step, all remaining sampling are interpolated. When being compared to the batch LTE method, computational complexity can be reduced by more than a magnitude. Yet it must be also clear that the method provides only an approximate solution of either (2.4) or the optimal solution presented in [16].

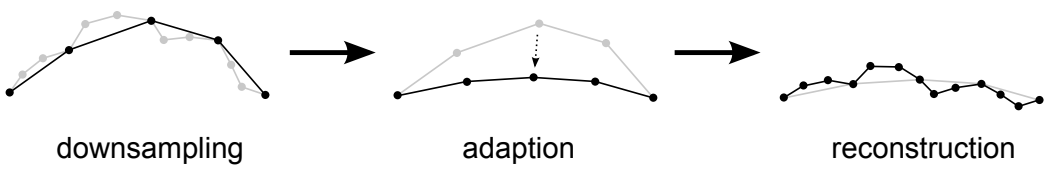


Figure 2.4: Multiresolution approach overview

Path downsampling

The goal of path downsampling is to obtain a subset of support sampling points $\mathbf{P}' = [\mathbf{p}'_1, \mathbf{p}'_2, \ldots, \mathbf{p}'_p]^T \in \mathbb{R}^{p \times m}$, $p \ll n$ subject to

$$\mathbf{P}' \in \mathbf{P}, \tag{2.29}$$

$$\min \sum_{i=2}^{p-1} (\|\mathbf{p}'_i - \mathbf{p}'_{i-1}\|_2 - \|\mathbf{p}'_{i+1} - \mathbf{p}'_i\|_2)^2. \tag{2.30}$$

The two conditions allow to interpolate the remaining sampling points during the reconstruction step without further coordinate transformations. In addition, they enforce a similar distance between subsequent sampling points ($\|\mathbf{p}'_i - \mathbf{p}'_{i-1}\|_2 = \|\mathbf{p}'_{i+1} - \mathbf{p}'_i\|_2$) which is necessary as combinatorial mesh Laplacians do not take into account the underlying geometry of the graph and thus rely on a regular mesh structure for a good approximation [37, 293, 311]. As it is

$$\mathfrak{F}(\Delta f(x)) = \mathfrak{F}(\nabla^2 f(x)) \propto u^2 F(u),$$

with $\mathfrak{F}$ as the Fourier transform from the spatial domain $f(x)$ to the frequency domain $F(u)$, the Laplace operator is heavily influenced by high-frequency noise. To increase robustness, the path has to be smoothed in the spatial or frequency domain using a spatial moving average (SMA) filter respectively a fast Fourier transform (FFT). In the limit, this will result in a regular mesh structure $\|\mathbf{p}'_i - \mathbf{p}'_{i-1}\|_2 = \|\mathbf{p}'_{i+1} - \mathbf{p}'_i\|_2 = const$. If additional fixed positional constraints are defined, the corresponding sampling points have to be included in $\mathbf{P}'$.

Path adaptation

After all support sampling points are determined, the downsampled trajectory $\mathbf{P}'$ is adapted. Similar to [280], an iterative two-staged approach is used. For the remainder of this section, support sampling points of iteration it are marked with $\mathbf{P}'_{it}$ and the local coordinates of the support sampling points with $\mathbf{\Delta}'_{it}$. In the first step, the resulting LTE equation system is solved for $\mathbf{P}'_{it+1}$ as

$$\mathbf{P}'_{it+1} = \begin{bmatrix} \mathbf{L} \\ \Omega\bar{\mathbf{P}} \end{bmatrix}^+ \begin{bmatrix} \mathbf{\Delta}'_{it} \\ \Omega\bar{\mathbf{C}} \end{bmatrix}.$$

In the second step, the elements of $\mathbf{\Delta}'_{it}$ are updated individually for every support sampling using Arun's method based on $\mathbf{P}'_{it}$ and $\mathbf{P}'_{it+1}$, resulting in $\mathbf{\Delta}'_{it+1}$. After l iterations, this results in the downsampled path $\mathbf{P}'_l = [\mathbf{p}'_{l,1}, \ldots, \mathbf{p}'_{l,n'}]^T$ with associated rotation matrix $\mathbf{R}'_k$, $k = \{1, 2, \ldots, n'\}$, measuring the rotation between $\mathbf{P}'_1$ and $\mathbf{P}'_l$ based on Arun's method.

Path reconstruction

In the last step all remaining sampling points are reconstructed, resulting in the upsampled deformed path $\mathbf{P}_s$ with a similar number of sampling points as the undeformed path

$\mathbf{P}$. Let the k-th path segment of $\mathbf{P}_s$ - named $\mathbf{P}_{s,k}$ - be defined as the set of all sampling points between $\mathbf{p}'_{l,k}$ and $\mathbf{p}'_{l,k+1}$, corresponding to $\mathbf{p}_{i,s}$ and $\mathbf{p}_{j,s}$ of the upsampled deformed path. In this context "between" refers to the graph structure and not the spatial domain. During the reconstruction step, the position of all sampling points of $\mathbf{P}_{s,k}$ are calculated through LTE as

$$\mathbf{P}_{s,k} = \begin{bmatrix} \mathbf{L}_{s,k} \\ \Omega\bar{\mathbf{P}}_{s,k} \end{bmatrix}^{+} \begin{bmatrix} \boldsymbol{\Delta}_{s,k} \\ \Omega\bar{\mathbf{C}}_{s,k} \end{bmatrix},$$

with $\mathbf{L}_{s,k}$ as the Laplacian matrix for the k-th trajectory segment, $\boldsymbol{\Delta}_{s,k}$ containing the rotated local coordinates and suitable boundary constraints encoded in $\bar{\mathbf{P}}_{s,k}, \bar{\mathbf{C}}_{s,k}$.

The matrix $\mathbf{L}_{s,k}$ is a submatrix of $\mathbf{L}$ with similar structure. The boundary constraints in $\bar{\mathbf{P}}_{s,k}, \bar{\mathbf{C}}_{s,k}$ are obtained by fixing the first and last sampling point of every trajectory segment.The elements of $\boldsymbol{\Delta}_{s,k}$ are calculated by linearly interpolating the differential coordinates of the path segment based on $\mathbf{R}'_k$ and $\mathbf{R}'_{k+1}$. In 2D and 3D the local coordinates of any other sampling point with index $i < o < j$ are either linearly interpolated based on an axis/angle representation of the rotational matrices $\mathbf{R}'_k$ or through SLERP/NLERP based on a quaternion representation. Using SLERP [272], the local coordinate $\boldsymbol{\delta}_o \in \boldsymbol{\Delta}$ is rotated as

$$\mathbf{q}_o = \frac{\mathbf{q}_k \sin\left(\frac{j-o}{j-i}\right) + \mathbf{q}_{k+1} \sin\left(\frac{o-i}{j-i}\right)}{\sin(\Theta)},$$
$$\boldsymbol{\delta}_{o,s} = \mathbf{q}_o \boldsymbol{\delta}_o \mathbf{q}_o^{-1},$$

with $\mathbf{q}_k, \mathbf{q}_{k+1}$ as the quaternion representation of $\hat{\mathbf{R}}_k, \hat{\mathbf{R}}_{k+1}$, $\mathbf{q}_l$ as the interpolated quaternion and $\boldsymbol{\delta}_{o,s} \in \boldsymbol{\Delta}_{s,k}$ as the rotated Laplacian coordinate.

Path reconstruction in 3D

The multiresolution approach is suitable for an arbitrary dimension and mainly limited by finding a interpolation scheme for the Laplacian coordinates in higher dimensions. In 3D computational complexity can be further reduced by using affine transformations for the reconstruction step instead of LTE. Affine transformations are discussed in detail in [245] and recapitulated only briefly here. For a path consisting of multiple sampling points $\mathbf{p}_i$ they are of the general form

$$\tilde{\mathbf{p}}_i = \mathbf{M}\mathbf{p}_i + \mathbf{w},$$

with $\mathbf{M} \in \mathbb{R}^{m \times m}, \mathbf{w} \in \mathbb{R}^m$. Boundary constraints for discrete paths ensuring C^q continuity at the beginning/end of the path are incorporated by fixing the first respectively last $q+1$ sampling points. Hence for C^0 continuity it is

$$\tilde{\mathbf{p}}_1 = \mathbf{M}\mathbf{p}_1 + \mathbf{w},$$
$$\tilde{\mathbf{p}}_n = \mathbf{M}\mathbf{p}_n + \mathbf{w},$$

and for C^1 continuity it is in addition

$$\tilde{\mathbf{p}}_2 = \mathbf{M}\mathbf{p}_2 + \mathbf{w},$$
$$\tilde{\mathbf{p}}_{n-1} = \mathbf{M}\mathbf{p}_{n-1} + \mathbf{w},$$

resulting in the linear equation system

$$\mathbf{V}\mathbf{b} = \mathbf{c}, \tag{2.31}$$

with $\mathbf{V} \in \mathbb{R}^{4m\times(m^2+m)}$, $\mathbf{b} \in \mathbb{R}^{m^2+m}$ containing the elements of $\mathbf{M},\mathbf{w}$ and $\mathbf{c} = \left[\tilde{\mathbf{p}}_1^T, \tilde{\mathbf{p}}_2^T, \tilde{\mathbf{p}}_{n-1}^T, \tilde{\mathbf{p}}_n^T\right]^T \in \mathbb{R}^{4m}$. Thus C^q continuity is only achieved if there are less or exactly as many boundary conditions as free variables in $\mathscr{M}$. As such for C^0 continuity at least one dimension and for C^1 continuity at least three dimensions are required. Affine transformations are advantageous from a computational perspective as only a small-sized linear equation system has to be solved for every path segment. Yet the method suffers from instabilities if the matrix $\mathbf{V}$ becomes singular. To prevent this, the four boundary sampling points must span a three-dimensional space. By defining the condition numbers $\kappa(\mathbf{V}_k)$ and $\kappa(\mathbf{V}_{k-1})$ based on (2.31) for every path segment, (2.29)-(2.30) are reformulated as

$$\begin{aligned}&\mathbf{P}' \in \mathbf{P},\\ &\min f_1 \sum_{i=2}^{n'-1}(\|\mathbf{p}'_i - \mathbf{p}'_{i-1}\|_2 - \|\mathbf{p}'_{i+1} - \mathbf{p}'_i\|_2)^2 + f_2 \sum_{i=1}^{n'-1}\kappa(\mathbf{V}_k),\end{aligned}$$

with constants f_1 and f_2.

Computational complexity comparison

Simulations compare the computational complexity of the different approaches. Shown in Fig. 2.5 is the processing time for a single deformation step over the number of path sampling points n, performed on a Intel Core2Duo T7500 2.2GHz CPU with 4GB RAM with Matlab R2010a running under Windows 7. Five different approaches are evaluated, an ARAP deformation which is a state-of-the-art approach from literature and the multiresolution approach presented in this section with two different methods for the reconstruction step. As the reconstruction step constitutes the computational bottleneck for all multiresolution approaches, the different downsampling/adaptation methods are not evaluated more in detail. All methods are compared to a non-multiresolutional LTE approach, either in its original version or using a spline representation. The main properties of all approaches are compared in Tab. 2.1, indicating whether it is a multiresolution approach, able to handle nonlinear deformation effects, applicable in an arbitrary number of dimensions and the section is presented in. It is visible that all algorithms that do not

name	multires.	nonlin. effects	general applic.	presented in
ARAP optimization		✓	✓	Sec. 2.5
Laplacian reconstruction	✓	✓	✓	Sec. 2.7
Affine reconstruction	✓	✓		Sec. 2.7
Spline approach				Sec. 2.6.2
Original approach			✓	Sec. 2.4

Table 2.1: Features of different retargeting approaches

consider nonlinear deformation effects (spline approach, original approach) outperform the other approaches. For large trajectories ($n \approx 10^5$) affine reconstruction and the spline approach have roughly the same computational complexity, yet affine reconstruction is only applicable in 3D. All presented approaches outperform the current state-of-the-art approach (ARAP optimization), either because they are based on a multiresolution approach or because they do not take nonlinear deformation effects into account. Yet one has to remember that all other approaches only approximate the optimal solution given by the ARAP optimization. For a given path consisting of n sampling points in an m-

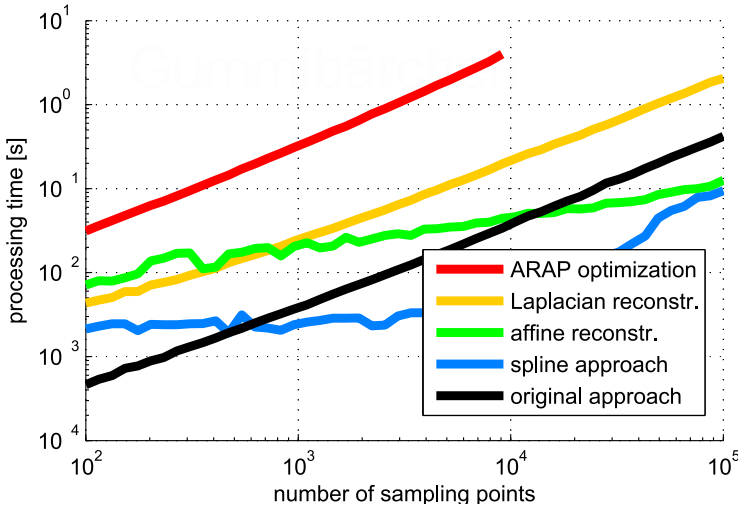


Figure 2.5: Processing time comparison between different LTE methods

dimensional space both spline and original approach have a computational complexity of $\mathcal{O}(nm)$. For the spline approach the transformation from continuous to discrete domain scales linearly with n and m. The original approach scales linearly with nm due to a sparse linear equation system for every dimension. All other approaches have a computational complexity of $\mathcal{O}(nm^3)$ because they rely at some point on Arun's method requiring a SVD on a $m \times m$-matrix and scale linearly with the number of sampling points n.

Spatial comparison

Comparisons between the approaches are also performed in the spatial domain. For this purpose a helix-shaped sinusoidal path is deformed by imposing four positional constraints, see Fig. 2.6. The methods compared are the original approach (Sec. 2.4), ARAP optimization (Sec. 2.5) and multiple downsampling/reconstruction combinations for the multiresolution approach. The spline approach is not shown as its path coincides with the original approach. It is visible how the affine reconstruction method without modified downsampling is mathematically unstable and differs strongly from the original path. Shown in the bottom bar graphs are normalized similarity values for all other methods, indicating how the original approach minimizes E_1 and the ARAP optimization minimizes E_2.

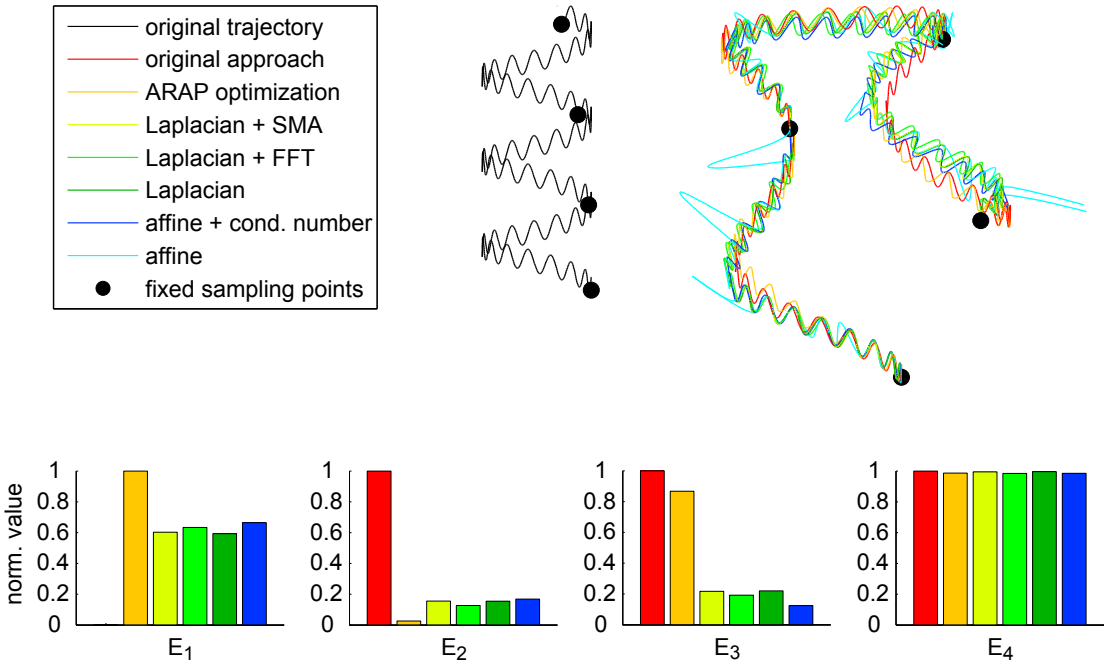


Figure 2.6: Spatial comparison between different LTE methods. Top: Spatial extension of different methods. Bottom: Corresponding similarity measures E_1-E_4

2.8 Cooperative manipulation of multiple agents through LTE

This section describes how LTE can be used to adapt the movement of multiple agents/trajectories in a coordinated manner which is an essential requirement for many collaborative scenarios like bimanual manipulation or coordinated swarm behavior. The key idea is to deform more than one trajectory in a systematic way by imposing additional constraints between the trajectories.

Given a set of trajectories $\mathbf{P}_1 = [\mathbf{p}_{1,1}, \mathbf{p}_{1,2}, \ldots, \mathbf{p}_{1,n}]^T, \ldots, \mathbf{P}_l = [\mathbf{p}_{l,1}, \mathbf{p}_{l,2}, \ldots, \mathbf{p}_{l,n}]^T \in \mathbb{R}^{n \times m}$, the corresponding LTE equation system is written for the resulting trajectories $\mathbf{P}_{s,1}, \ldots, \mathbf{P}_{s,l} \in \mathbb{R}^{n \times m}$ as

$$\begin{bmatrix} \mathbf{L} & & \\ & \ddots & \\ & & \mathbf{L} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{s,1} \\ \vdots \\ \mathbf{P}_{s,l} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta}_1 \\ \vdots \\ \boldsymbol{\Delta}_l \end{bmatrix}.$$

Under the influence of additional constraints, it is rewritten as

$$\begin{bmatrix} \mathbf{L} & & \\ & \ddots & \\ & & \mathbf{L} \\ \boldsymbol{\Omega}_c \bar{\mathbf{P}}_1 & \ldots & \boldsymbol{\Omega}_c \bar{\mathbf{P}}_l \end{bmatrix} \begin{bmatrix} \mathbf{P}_{s,1} \\ \vdots \\ \mathbf{P}_{s,l} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta}_1 \\ \vdots \\ \boldsymbol{\Delta}_l \\ \boldsymbol{\Omega}_c \bar{\mathbf{C}} \end{bmatrix},$$

with the matrices $\bar{\mathbf{P}}_1, \ldots, \bar{\mathbf{P}}_l, \bar{\mathbf{C}}$ accounting for the constraints and the weighting matrices $\boldsymbol{\Omega}_c$.

Fig. 2.7 shows a toy scenario in which two respectively three agents have to maintain a defined spatial distance defined by wide red lines while circumnavigating two obstacles (black cylinders). Shown are both undeformed trajectories without obstacles and deformed trajectories in the presence of obstacles. For two agents, the corresponding equation system is given by

$$\begin{bmatrix} \mathbf{L} & 0 \\ 0 & \mathbf{L} \\ \boldsymbol{\Omega}_c \bar{\mathbf{P}}_- & \boldsymbol{\Omega}_c \bar{\mathbf{P}}_+ \end{bmatrix} \begin{bmatrix} \mathbf{P}_{s,1} \\ \mathbf{P}_{s,2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta}_1 \\ \boldsymbol{\Delta}_2 \\ \boldsymbol{\Omega}_c \bar{\mathbf{C}} \end{bmatrix},$$

with the definition of the matrices $\boldsymbol{\Omega}_c, \bar{\mathbf{P}}_-, \bar{\mathbf{P}}_+ \in \mathbb{R}^{n \times n}$ and $\bar{\mathbf{C}}$ as

$$\begin{aligned} \boldsymbol{\Omega}_c &= \mathrm{diag}(\omega_c, \ldots, \omega_c), \\ \bar{\mathbf{P}}_- &= \mathrm{diag}(-1, \ldots, -1), \\ \bar{\mathbf{P}}_+ &= \mathrm{diag}(1, \ldots, 1), \\ \bar{\mathbf{C}}_{\{i:\}} &= \mathbf{p}_{1,i} - \mathbf{p}_{2,i}, \end{aligned}$$

with weighting factor ω_c, thus specifying the desired distance between sampling points of the two trajectories with similar index i. Under the condition of identical timings $t_i(\mathbf{p}_{1,i}) = t_i(\mathbf{p}_{2,i})$, the two trajectories maintain a defined spatial distance $\mathbf{d}_i \approx \mathbf{p}_{1,i} - \mathbf{p}_{2,i}$ at time instance i. The difference between the constraints in (2.7) and $\bar{\mathbf{P}}_-, \bar{\mathbf{P}}_+$ is that the first accounts for the spatial distance of two subsequent sampling points of the same trajectory whereas the latter specifies a spatial distance between two sampling points of different trajectories. As the trajectories of both agents are coupled through $\bar{\mathbf{C}}$, it is sufficient to specify positional constraints only for a single agent to deform both trajectories. When extending the approach to three agents, it becomes

$$\begin{bmatrix} \mathbf{L} & 0 & 0 \\ 0 & \mathbf{L} & 0 \\ 0 & 0 & \mathbf{L} \\ \boldsymbol{\Omega}_c \bar{\mathbf{P}}_- & \boldsymbol{\Omega}_c \bar{\mathbf{P}}_+ & 0 \\ \boldsymbol{\Omega}_c \bar{\mathbf{P}}_- & 0 & \boldsymbol{\Omega}_c \bar{\mathbf{P}}_+ \\ 0 & \boldsymbol{\Omega}_c \bar{\mathbf{P}}_- & \boldsymbol{\Omega}_c \bar{\mathbf{P}}_+ \end{bmatrix} \begin{bmatrix} \mathbf{P}_{s,1} \\ \mathbf{P}_{s,2} \\ \mathbf{P}_{s,3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta}_1 \\ \boldsymbol{\Delta}_2 \\ \boldsymbol{\Delta}_3 \\ \boldsymbol{\Omega} \bar{\mathbf{C}}_1 \\ \boldsymbol{\Omega} \bar{\mathbf{C}}_2 \\ \boldsymbol{\Omega} \bar{\mathbf{C}}_3 \end{bmatrix},$$

in order to maintain a fixed spatial distance between every three sampling points of the three trajectories with similar index. Positional constraints according to (2.5) are incorporated in a straightforward manner and not described more in detail at this point. The graph on the right side displays the spatial distance d_{ij} for the undeformed trajectory and $d_{ij,m}$ for the deformed trajectory between agents j and j over time. Through proper choice of the weighting matrices $\boldsymbol{\Omega}$ the distance between every two agents stays constant over time. The error due to the LS solution is negligible for practical purposes as it is in the range of 1e-10 m. One also observes how the trajectories of all agents are adapted when just fixing the position of a single agent. The figure displays only the most primitive case with a constant spatial distance and direction across multiple agents which can be further modified through proper choice of the additional constraints.

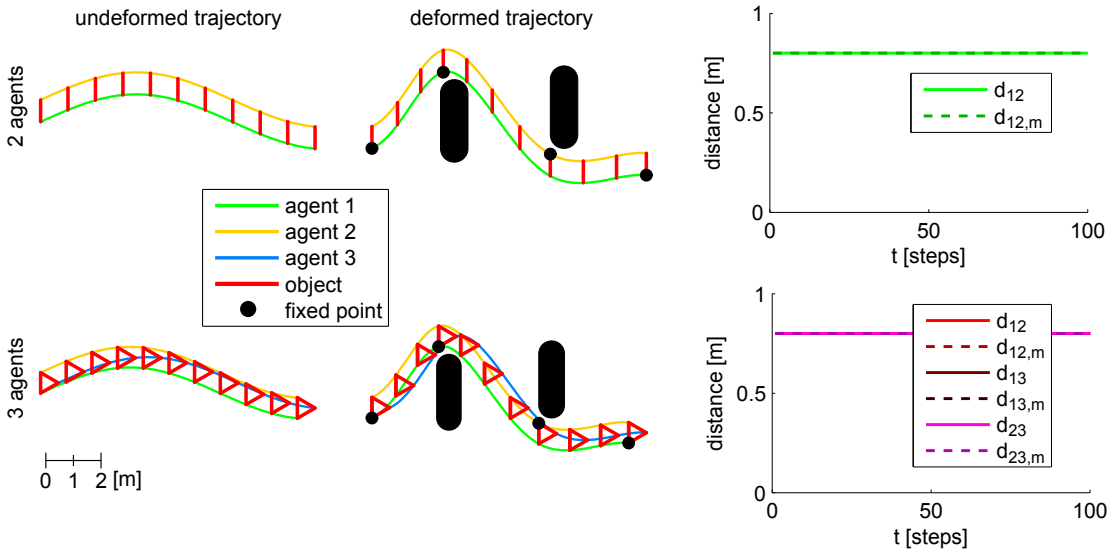


Figure 2.7: Cooperative manipulation involving two and three agents in the presence of obstacles. Left: Trajectory paths. Right: Distance between every two agents

2.9 LTE collision avoidance in cluttered environments

One of the most common problems during trajectory adaptation is circumnavigation around obstacles. This section presents different approaches for collision avoidance depending on the amount of obstacles. The first approach is directly deduced from the methods presented so far. By imposing positional constraints on specific sampling points, obstacles are avoided. The second approach makes use of variable positional constraints and constructs a repellent potential field around obstacles. The last approach derives an optimized solution through excessive planning, sampling the entire free search space until a valid, optimized solution is found. All methods have specific advantages and disadvantages. They also differ in their range of use. The first two approaches are best situated for gently constrained environment. Moreover it is assumed that all obstacles are known a priori and an efficient method measuring the distance between trajectory and obstacles is at hand. On the other hand the planning method only requires a collision detection scheme measuring whether an intersection of the trajectory with an obstacle happens. It does not rely on a precise distance measure between trajectory and obstacles. A drawback is the high computational complexity. Being only a minor modification of the original LTE method, the potential fields approach can be executed in real time to deform small trajectories. As the planning approach iteratively samples the free space to find a valid solution, there are no guaranteed bounds for convergence. The approach is also magnitudes slower, making offline calculation the only viable way to find a solution.

2.9.1 Collision avoidance through high-weighted positional constraints

This section grabs up the idea of deforming paths through positional constraints with a weight $\mathbf{\Omega} \gg 1$. Whereas the purpose of previous applications was to display the capacity of LTE during path adaptation, it can be also used to avoid obstacles in a systematic and efficient way. When using high-weighted positional constraints, they can be combined with the spline-based deformation approach as described in Sec. 2.6.2 for a fast collision avoidance scheme. By splitting a trajectory x up into smaller trajectory segments $x_1, x_2, \ldots, x_p$ such that the last point of trajectory segment x_i coincides with the first point of trajectory segment x_{i+1}, convex hulls are calculated for every trajectory segment. If the convex hull of the i-trajectory segment penetrates an obstacle with penetration depth $\mathbf{d}_i$, both first and last point $\mathbf{p}_{i,1}, \mathbf{p}_{i,l}$ of the trajectory segment are deformed through positional constraints as

$$\mathbf{p}_{i,1,s} = \mathbf{p}_{i,1} + \mathbf{d}_i,$$
$$\mathbf{p}_{i,l,s} = \mathbf{p}_{i,l} + \mathbf{d}_i.$$

As the trajectory gets deformed through the shifting operation, the shape of the new convex hull of x_i differs from the old one which can require multiple shifting operations until collision avoidance is guaranteed. In addition due to the coupling between two trajectory segments ($\mathbf{p}_{i,1} = \mathbf{p}_{i-1,l}$) the deformation has to take into account neighboring segments. A modified approach results in

$$\mathbf{p}_{i,1,s} = \mathbf{p}_{i,1} + \eta \max(\mathbf{d}_i, \mathbf{d}_{i-1}),$$
$$\mathbf{p}_{i,l,s} = \mathbf{p}_{i,l} + \eta \max(\mathbf{d}_i, \mathbf{d}_{i+1}),$$

with constant $\eta > 1$. Obstacle avoidance based on convex hulls is a very efficient and fast method. When bounding the convex hull from above as described in (2.28), the term $conv(\{1\}) \bigoplus conv(\{2\})$ can be quickly calculated. By using splines for the deformation process, $conv(\{2\})$ is given by the control points of each Bernstein polynomial which is faster than calculating it through a conventional, iterative scheme. The convex hull of the undeformed trajectory $conv(\{1\})$ is calculated offline as it does not change during deformation.

2.9.2 Collision avoidance through low-weighted positional constraints

A second method avoids collisions with obstacles by creating a repulsive force field around each obstacle while maintaining the original path's shape. When solving the LS solution (2.4), the weighting matrix $\mathbf{\Omega}$ is generally chosen as $\mathbf{\Omega} \gg 1$. This way constraints are prioritized when calculating the LS solution. In case only positional constraints are used, this results in a negligible error $\mathbf{p}_i - \mathbf{c}_{i0} \approx$. In contrast to that, this section considers low-weighted positional constraints with $\mathbf{\Omega} \approx 1$ or $\mathbf{\Omega} \ll 1$, resulting in a non-negligible positional error. For $\mathbf{L} \neq \mathbf{L}_0$ the resultant behavior is best described by a "force", pushing

or pulling a sample point of the path in a certain direction without pinning it to a specific position. By imposing low-weighted position constraints on all sampling points of the path and smoothly varying the constraints between subsequent sampling points, a continuous "force field" is imitated. The presented approach consists of three superimposed forces:

- A repulsive force for obstacle avoidance
- An attractive force for trajectory convergence towards its original position
- The Laplacian coordinates maintaining the local shape of the path

Unmodified approach

For a single obstacle Ψ the vector $\mathbf{d}_i$ accounts for the shortest distance between the obstacle and a sampling point $\mathbf{p}_i$ of the path, see Fig. 2.8. The repulsive force is then exerted in direction of $\mathbf{d}_i$ in order to prevent a collision of the path with any obstacle. It is desired that the repulsive force becomes larger if the sampling point approaches the obstacle. This is achieved by setting the positional constraint defined by $\bar{\mathbf{P}}_1$ and $\bar{\mathbf{C}}$ as

$$\begin{aligned} \bar{\mathbf{C}}_{1\{i:\}} &= \beta_w \left(\mathbf{p}_i + \alpha_w \frac{\mathbf{d}_i}{\|\mathbf{d}_i\|_2^{\gamma_w}} \right), \\ \bar{\mathbf{P}}_1 &= \mathrm{diag}(\beta_w, \ldots, \beta_w). \end{aligned} \tag{2.32}$$

The constants α_w, β_w and γ_w act as task-specific tuning parameters. A parameterization of $\gamma_w = 3$ for example lets the repulsive force decrease quadratically over the distanca $\mathbf{d}_i$, this resembling the force of an electric point charge in 3D. In a similar fashion the attractive force is described by a positional constraint of the form

$$,\bar{\mathbf{P}}_2 = \mathrm{diag}(\phi_w, \ldots, \phi_w),$$

pulling a sampling point $\mathbf{p}_i$ back to its original position $\mathbf{p}_{i,o}$ of the undeformed path. The constant ϕ_w adjusts the strength of the attractive force. To account for both repellent and attracting force, the terms in (2.4) accounting for the constraints are modified as

$$\mathbf{\Omega}\bar{\mathbf{C}} = \begin{bmatrix} \bar{\mathbf{C}}_1 \\ \bar{\mathbf{C}}_2 \end{bmatrix}, \quad \mathbf{\Omega}\bar{\mathbf{P}} = \begin{bmatrix} \bar{\mathbf{P}}_1 \\ \bar{\mathbf{P}}_2 \end{bmatrix}.$$

Modified approach

There are two problems associated with the unmodified approach. First the repulsive force acts individually on each sampling point $\mathbf{p}_i$ and not on the vertex between two subsequent sampling points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. The path thus can "slip through" or collide with an obstacle if two subsequent sampling points are either far apart or the obstacle is small. Another problem are instabilities if the amount of deformation gets too large. In

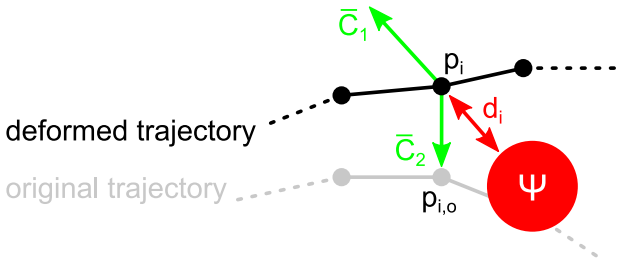


Figure 2.8: Collision avoidance through low-weighted positional constraints

this case the influence of all three forces may build up, leading to abrupt jumps of the deformed path. To overcome the first problem the shortest distance $\mathbf{d}_i$ is not calculated between the obstacle and every sampling point $\mathbf{p}_i$ but between the obstacle and the line segment $\mathbf{p}_i + \alpha_l(\mathbf{p}_{i+1} - \mathbf{p}_i)$, $\alpha_l \in [0,1]$, resulting in the shortest distance $\hat{\mathbf{d}}_i$. Hence (2.32) is reformulated as

$$\bar{\mathbf{C}}_{1\{i:\}} = \beta_w \left(\mathbf{p}_i + \alpha_w \frac{\hat{\mathbf{d}}_i}{\|\hat{\mathbf{d}}_i\|_2^{\gamma_w}} \right),$$
$$\bar{\mathbf{P}}_1 = \mathrm{diag}(\beta_w, \ldots, \beta_w).$$

The second problem is overcome by limiting both the attractive force and the force exerted by maintaining the Laplacian coordinates, thus letting the repulsive force become predominant when getting too close to an obstacle. The attractive force is limited by modifying $\bar{\mathbf{C}}_{2\{i:\}}$ as

$$\bar{\mathbf{C}}_{2\{i:\}} = \mathbf{p}_i + \epsilon_w \frac{\mathbf{p}_{i,o} - \mathbf{p}_i}{1 + \|\mathbf{p}_{i,o} - \mathbf{p}_i\|_2}, \tag{2.33}$$

with the constant ϵ_w specifying the force's upper bound. To limit the force exerted by the Laplacian coordinates, let $\mathbf{\Delta}_o$ and $\mathbf{\Delta}_d$ be the matrices with the Laplacian coordinates of the original and deformed path. By updating the matrix $\mathbf{\Delta}$ during each deformation step as

$$\mathbf{\Delta} = (1 - \zeta_w)\mathbf{\Delta}_o + \zeta_w \mathbf{\Delta}_d, \tag{2.34}$$

the influence of the Laplacian coordinates is varied smoothly via the constant $\zeta_w \in [0,1]$. For $\zeta_w = 0$ both modified and unmodified approach equal each other. For $\zeta_w = 1$ only the Laplacian coordinates of the deformed path are considered, effectively disabling the ability to maintain the local shape of the path. When comparing the modified and unmodified approach, the modified approach has a larger computational complexity, mainly due to the calculation of $\hat{\mathbf{d}}_i$. In addition it has a slower convergence rate back to the original trajectory in the absence of obstacles.

Relation to Elastic Strips

Both approaches are related to the Elastic Strips framework [39] as the forces can be split up into internal forces for maintaining the path's shape and external forces for path

deformation. Yet both their definition and purpose differs. Whereas LTE uses the discrete Laplace-Beltrami operator

$$\mathbf{F}_i^{int,L} = \boldsymbol{\delta}_i = \frac{\mathbf{p}_{i-1} - 2\mathbf{p}_i + \mathbf{p}_{i+1}}{-2}, \quad i = 2, 3, \ldots, n-1,$$

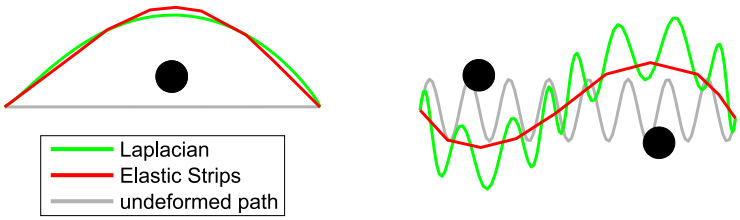
to describe the internal force $\mathbf{F}_i^{int,L}$, Elastic Strips rely on a heuristic definition for the internal force $\mathbf{F}_i^{int,E}$ as

$$\mathbf{F}_i^{int,E} = k_c \left(\frac{d_{i-1}}{d_{i-1} + d_i} (\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) - (\mathbf{p}_i - \mathbf{p}_{i-1}) \right), \quad i = 2, 3, \ldots, n-1, \tag{2.35}$$

with $d_i = \|\mathbf{p}_{i+1} - \mathbf{p}_i\|_2$. The external force $\mathbf{F}_i^{ext,E}$ is defined as

$$\mathbf{F}_i^{ext,E} = \begin{cases} k_r (d_0 - \|\mathbf{d_i}\|) \frac{\mathbf{d_i}}{\|\mathbf{d_i}\|} & \text{if } \|\mathbf{d_i}\| < d_0, \\ 0 & \text{otherwise.} \end{cases}$$

Whereas Elastic Strips try to maintain the shortest possible path in task space, LTE tries to maintain the original shape of the path. If the undeformed path is a straight line, the result after deformation is roughly the same, see Fig. 2.9. Yet Elastic Strips cannot be applied to non-straight paths as they'll always converge to a straight path in the absence of obstacles. Both methods tackle the problem of large deformations by modifying the internal forces. Whereas Elastic Strips use a modified minimum-distance formulation (2.35) that shares common properties with curvature based methods, LTE scales the internal forces as described in (2.33) and (2.34). Elastic Strips are advantageous from a computational point of view in two ways. First they only add sampling points (robot configurations) to the path when necessary, keeping the overall number low whereas LTE always considers all sampling points. It also relies on a matrix inversion whereas Elastic Strips are calculated through a computationally more efficient gradient descent approach. On the other hand the LS approach of LTE converges faster as an optimal solution is calculated at every time step.



LTE	Elastic Strips
$\alpha_w = 0.05$	$k_c = 0.05$
$\beta_w = 0.05$	$k_r = 0.02$
$\gamma_w = 2$	$d_0 = 3$
$\epsilon_w = 0$	
$\phi_w = 0.002$	
$\zeta_w = 0$	
$\omega = 1e6$	

Figure 2.9: Comparison between Elastic Strips and LTE. Left: Straight path. Right: Sinusoidal path

2.9.3 Collision avoidance through incremental planning using Laplacian-RRT*

The approach described in this section shifts focus from reactive collision avoidance strategies that are executed in real time to offline path planning methods which must be calculated before execution. Whereas until now always the entire path is deformed when avoiding obstacles, such methods are only applicable in gently constrained environments, this section presents an incremental approach based upon the RRT* algorithm that grows a search tree following the shape of the original path to connect to positions in task space.

Overview

Path planning in constrained environments is a well elaborated problem in robotic applications. A recent trend goes towards sampling-based planning, focusing not on analytical optimal solutions but on the exploitation of today's available computing power. One of the most successful incremental sampling-based planners is the RRT* algorithm. The algorithm is used to connect a start and end state via a path by growing a connected tree with local optimal properties while avoiding obstacles and sampling the free space randomly when expanding the tree. It consists of two different, interlocking steps. In the first step a new sampling node is added to a search tree according to a set of rules. In the limiting case the search tree spans across the entire search space, reaching every state from a given start state. A feasible path connecting the desired start and end state is then found by backtracking the nodes along the tree. Yet the resulting solution is in general not optimal. Thus the tree is rewired in the second step on a local scale based upon a given cost function to create a tree with optimal local properties. It is shown in [142] that the resulting tree is also optimal on a global scale in the limiting case, providing an optimal path. Its pseudocode is given in Alg. 1-3. Its core components are

Algorithm 1: RRT* $(\mathcal{V}, \mathcal{E}, N)$

for $i = 1, \ldots, N$ **do**
  $y_{rand} \leftarrow$ `Sample`;
  $y_{nearest} \leftarrow$ `Nearest`$(\mathcal{V}, y_{rand})$;
  $y_{new} \leftarrow$ `Approach`$(y_{nearest}, y_{rand})$;
  if `CollisionFree`$(y_{nearest}, y_{new})$ **then**
    $Y_{near} \leftarrow$ `Near`$(\mathcal{V}, y_{new})$;
    $(\mathcal{V}, \mathcal{E}) \leftarrow$ `Parent`$(\mathcal{V}, \mathcal{E}, Y_{near}, y_{new})$;
    $(\mathcal{V}, \mathcal{E}) \leftarrow$ `Rewire`$(\mathcal{V}, \mathcal{E}, Y_{near}, y_{new})$;
return $(\mathcal{V}, \mathcal{E})$

- `Cost`: The optimal cost of a state y based on the underlying distance metric $dist(y, y')$ between two states y and y' in configuration space C.

Algorithm 2: `Parent`$(\mathcal{V}, \mathcal{E}, Y_{near}, y_{new})$

$\mathcal{V} \leftarrow \mathcal{V} \cup y_{new}$;
minCost $\leftarrow \infty$; $y_{min} \leftarrow$ NULL; $\sigma_{min} \leftarrow$ NULL;
for $y_{near} \in Y_{near}$ **do**
  $\sigma \leftarrow$ `Steer`(y_{near}, y_{new});
  if `Cost`(x_{near}) + `Cost`(σ) < minCost **and** `CollisionFree`(σ) **then**
    minCost $\leftarrow$ `Cost`(y_{near}) + `Cost`(σ);
    $y_{min} \leftarrow y_{near}$; $\sigma_{min} \leftarrow \sigma$;
$\mathcal{E} \leftarrow \mathcal{E} \cup \{y_{near}, y_{new}\}$;
return $(\mathcal{V}, \mathcal{E})$

Algorithm 3: `Rewire`$(\mathcal{V}, \mathcal{E}, Y_{near}, y_{new})$

for $y_{near} \in Y_{near}$ **do**
  $\sigma \leftarrow$ `Steer`(y_{new}, y_{near});
  if `Cost`(y_{new}) + `Cost`(σ) < `Cost`(y_{near}) **and** `CollisionFree`(σ) **then**
    $y_{parent} \leftarrow$ `Parent`(y_{near});
    $\mathcal{E} \leftarrow \mathcal{E} \setminus \{y_{parent}, y_{near}\}$;
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{y_{new}, y_{near}\}$;
return $(\mathcal{V}, \mathcal{E})$

- `Near`/`Nearest`: Returns a set of states $Y \in \mathcal{V}$ resp. the nearest state $y_{nearest}$ from the vertices $\mathcal{V}$ of the graph $(\mathcal{V}, \mathcal{E})$ within a certain radius
- `Steer`: Gives a path σ connecting two states y and y'.
- `Approach`: Returns a state that is closer to y' from y by a predefined amount α_r.
- `CollisionFree`: Checks if the path σ connecting two states y and y' lies entirely in the non-colliding configuration space $C_{free} \subseteq C$.
- `Sample`: Performs independent uniformly random sampling of C_{free}.

When expanding the tree graph, i.e. adding a leaf node to the tree, the RRT* algorithm performs as follows: In a first step a random state y_{rand} is sampled from the free configuration space. After having found the closest node $y_{nearest}$, the random state is approached by a predefined amount α_r, resulting in the new state y_{new}. Until now the approach is similar to the RRT algorithm and aims at sampling the entire free configuration space C_{free}. To create a tree with local optimal properties, its nodes are rewired dynamically. If the path between $y_{nearest}$ and y_{new} is non-colliding, the `Parent` function selects the parent node with minimal cost from Y_{near} for y_{new}. The function `Rewire` does the same for leaf nodes in Y_{near} and rewires the graph accordingly.

Distance metric

A key feature of RRT* is the underlying distance metric which determines how far two states are apart from each other. In general it is either chosen to be the Euclidean distance in Cartesian space or time, resulting in shortest paths or time-optimal trajectories. For being combined with LTE, the distance metric is defined as the residual of the LTE solution. Without loss of generality it is defined as the weighted velocity and acceleration deviation between two sampling points $\mathbf{p}_i \in \mathbf{P}$ of the reference trajectory and $\mathbf{p}_{i,s} \in \mathbf{P}_s$ of the tree as

$$dist(\mathbf{p}_i, \mathbf{p}_{i,s}) = \omega_1 \|\mathbf{c}_{i1,s} - \mathbf{c}_{i1}\|_F^2 + \omega_2 \|\mathbf{c}_{i2,s} - \mathbf{c}_{i2}\|_F^2, \tag{2.36}$$

with weighting factors $\omega_1 \in \mathbf{\Omega}_1, \omega_2 \in \mathbf{\Omega}_2$ and $\mathbf{c}_{i1,s}, \mathbf{c}_{i1}, \mathbf{c}_{i2,s}, \mathbf{c}_{i2}$ as defined in Sec. 2.4. This representation holds in case of equitemporally spaced sampling points. When comparing two paths with multiple sampling points, one has to take the sum over n sampling points. In addition, every tree path $\mathbf{P}_s$ has to pass at least the first and last sampling point. This is reformulated as

$$\begin{aligned} d_{sum} &= \sum_{i=1}^{n} dist(\mathbf{p}_i, \mathbf{p}_{i,s}) = \sum_{i=1}^{n} \omega_1 \|\mathbf{c}_{i1,s} - \mathbf{c}_{i1}\|_F^2 + \omega_2 \|\mathbf{c}_{i2,s} - \mathbf{c}_{i2}\|_F^2, \\ &\text{s.t. } \bar{\mathbf{P}}\mathbf{P}_s = \bar{\mathbf{C}}. \end{aligned} \tag{2.37}$$

and solved through LTE for the unconstrained case as minimizing d_{sum} corresponds to solving (2.12). In order to apply the distance metric in (2.36) to RRT*, three modifications are necessary.

For the given distance metric the RRT* algorithm operates in the acceleration/velocity (configuration) space. As the acceleration is calculated through second order finite differences based on three subsequent sampling points $\{\mathbf{p}_{i+1}, \mathbf{p}_i, \mathbf{p}_{i-1}\} \in \mathbb{R}^{3n}$, the configuration space is $3n$-dimensional. Because the velocity is calculated based on $\{\mathbf{p}_i, \mathbf{p}_{i-1}\}$ and those two points are already included in the configuration space, adding the velocity does not change its dimension. When sampling the configuration space, asymptotic optimality is guaranteed [142]. Yet due to the high dimensionality of the configuration space the algorithm can become unfeasible slow. Another option is to operate in task space, defined by the sampling points $\mathbf{p}_i$. By associating every velocity/acceleration vector with an individual sampling point $\mathbf{p}_i, \mathbf{p}_{i,s}$ in task space, the dimensionality of the search space is reduced from $3n$ to n.

Another aspect are indexed sampling points. As comparing the local path properties is only reasonable for two sampling points $\mathbf{p}_i, \mathbf{p}_{i,s}$ with similar index along the trajectory, the distance metric is only defined in this case. Implicitly it is assumed that $\mathbf{P}$ and $\mathbf{P}_s$ consist of the same number of sampling points. Indexing the sampling points has another consequence: When modifying the connections between nodes in the tree as done by the `Parent` or `Rewire` function, it has to be assured that only nodes with subsequent sampling points are connected.

Last, the resulting tree is grown from nodes with lower index towards nodes with higher index in order to ensure the continuous increase of subsequent sampling points' indices, making it necessary to use backward finite differences when calculating $\ddot{\mathbf{p}}_i$ and $\ddot{\mathbf{p}}_{i,s}$.

All three modifications lead to a new distance metric

$$dist(y, y') = dist(\mathbf{p}_i, \mathbf{p}_{i,s}) = \omega_1 \|\mathbf{c}_{i1,s} - \mathbf{c}_{i1}\|_F^2 + \omega_2 \|\mathbf{c}_{i2,s} - \mathbf{c}_{i2}\|_F^2, \tag{2.38}$$

associating each state y, y' with a specific point $\mathbf{p}_i, \mathbf{p}_{i,s}$ in task space.

To account for all modifications, the pseudocode of the RRT* algorithm has to be modified as described in Alg. 4-6. To consider the index of all sampling points properly, a new variable $id_{rand} \in \{1, 2, \ldots, n\}$ is introduced. As a result, the `ParentId` function only connects the new node y_{new} to other nodes with index $id_{rand} - 1$. The `RewireId` works in a similar fashion and reconnects only other nodes with an index $id_{rand} + 1$ or $id_{rand} + 2$. Because the acceleration is calculated based on three subsequent sampling points, also nodes with index $id_{rand} + 2$ must be considered for the `RewireId` function. In addition, the function has to check every possible combination $\{\mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i\}$, $i \in \{id_{rand} + 1, id_{rand} + 2\}$ for better cost/collision and rewire when necessary. For better distinction between the original RRT* algorithm and the modified version, it is renamed as Laplacian-RRT*.

If the set Y_{near} consists of all vertices V, the asymptotic optimality property of RRT* still holds. Even if the task space has a lower dimension than the configuration space, all elements of the configuration space are uniquely calculated based on the position of the sampling points $\{\mathbf{p}_{i+1}, \mathbf{p}_i, \mathbf{p}_{i-1}\}$ in task space. Still, as indexed sampling points impose additional restrictions on which nodes to rewire, the algorithm has a slow convergence rate.

Algorithm 4: Laplacian-RRT* $(\mathcal{V}, E, N)$

```
for it = 1, ..., N do
    y_rand, id_rand ← Sample;
    y_nearest ← Nearest(V, y_rand, id_rand);
    y_new ← Approach(y_nearest, y_rand);
    if CollisionFree(y_nearest, y_new) then
        Y_near ← V;
        (V, E) ← ParentId(V, E, Y_near, y_new, id_rand);
        (V, E) ← RewireId(V, E, Y_near, y_new, id_rand);
return (V, E)
```

Task space bias

A task space bias [270, 271] is introduced to increase the convergence rate and reduce computational complexity. Rather than sampling the entire task space randomly, the growth of the tree is biased towards solutions with a smaller cost function d_{sum}. Whereas this provides a fast and good solution, it also limits the amount of exploration. The task space bias accelerates calculations in two ways. First it increases the speed at which a valid solution is found. The stronger the task space bias is, the closer the valid solution will be to the optimal one. Second it allows one to use a standard nearest neighbor

Algorithm 5: `ParentId`$(\mathcal{V}, E, Y_{near}, y_{new}, id_{rand})$

$\mathcal{V} \leftarrow \mathcal{V} \cup y_{new}$;
minCost $\leftarrow \infty$; $y_{min} \leftarrow$ NULL; $\sigma_{min} \leftarrow$ NULL;
for $y_{near} \in Y_{near}$ **do**
  if $id(y_{near}) = id_{rand} - 1$ **then**
    $\sigma \leftarrow$ `Steer`(y_{near}, y_{new});
    if `Cost`(y_{near}) + `Cost`(σ) < minCost **and** `CollisionFree`(σ) **then**
      minCost $\leftarrow$ `Cost`(y_{near}) + `Cost`(σ);
      $y_{min} \leftarrow y_{near}$; $\sigma_{min} \leftarrow \sigma$;
$E \leftarrow E \cup \{y_{near}, y_{new}\}$;
return $(\mathcal{V}, E)$

Algorithm 6: `RewireId`$(\mathcal{V}, E, Y_{near}, y_{new}, id_{rand})$

for $y_{near} \in Y_{near}$ **do**
  if $id(y_{near}) = id_{rand} + 1$ **or** $id(y_{near}) = id_{rand} + 2$ **then**
    $i = id_{near}$;
    Check any possible combination $\{\mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i\}$ for lower cost/collision and rewire when necessary
return $(\mathcal{V}, E)$

search in task space when determining the set Y_{near}. It has been explained before that LTE minimizes the term d_{sum} in (2.37) for the unconstrained case. Mathematically, when expanding a tree branch $\mathbf{P}_b = [\mathbf{p}_{b,1}, \mathbf{p}_{b,2}, \ldots, \mathbf{p}_{b,l}]^T$ with another sampling point with index $l+1$, its optimal position $\hat{\mathbf{p}}_{b,l+1}$ is calculated as

$$\mathbf{P}_s = \begin{bmatrix} \mathbf{\Omega}_1 \mathbf{L}_1 \\ \mathbf{\Omega}_2 \mathbf{L}_2 \\ \mathbf{\Omega}\bar{\mathbf{P}} \end{bmatrix}^+ \begin{bmatrix} \mathbf{\Omega}_1 \mathbf{\Delta}_1 \\ \mathbf{\Omega}_2 \mathbf{\Delta}_2 \\ \mathbf{\Omega}\bar{\mathbf{C}} \end{bmatrix}$$

$$\hat{\mathbf{p}}_{b,l+1} = \mathbf{P}_{s\{l+1,:\}}$$

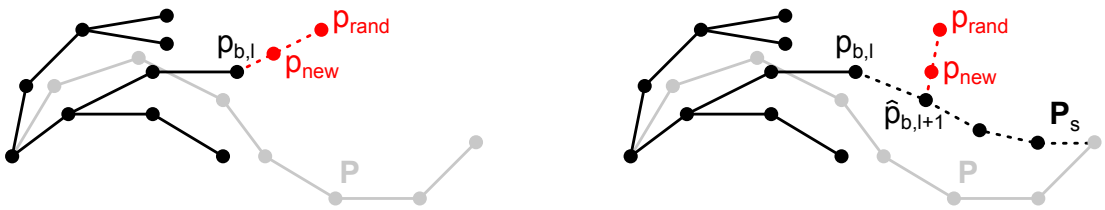


Figure 2.10: Task space bias. Left: Conventional approach. Right: Modified approach

with $\mathbf{\Omega} = \text{diag}(\omega, \ldots, \omega)$. The resulting vector $\hat{\mathbf{p}}_{b,l+1} - \mathbf{p}_{b,l}$ is used as an offset for the `Approach` function to bias the tree growth towards the optimal trajectory. Whereas a conventional approach results in

$$\mathbf{p}_{new} = \mathbf{p}_{b,l} + \alpha_r(\mathbf{p}_{rand} - \mathbf{p}_{b,l}),$$

the modified version becomes

$$\mathbf{p}_{new} = \hat{\mathbf{p}}_{b,l+1} + \alpha_r(\mathbf{p}_{rand} - \hat{\mathbf{p}}_{b,l+1}), \tag{2.39}$$

with the variable $\alpha_r \in [0, 1]$ accounting for the amount of bias, see Fig. 2.10.

Task space nearest neighbors

A computational bottleneck of all RRT-based algorithms is the need of a fast nearest neighbor search for tree expansion. Whereas one can define any distance metric $dist(x, x')$, the challenge is to find a nearest neighbor algorithm both being consistent with the chosen distance metric and reasonably fast. The LS solution depends quadratically on $\mathbf{L}_1$ and $\mathbf{L}_2$ and therefore also on the position of all sampling points $\mathbf{p}_i$. For a given branch $\mathbf{P}_b = [\mathbf{p}_{b,1}, \mathbf{p}_{b,2}, \ldots, \mathbf{p}_{b,l}]^T$ of the tree, the next sampling point's position $\mathbf{p}_{b,l+1}$ is split up as

$$\mathbf{p}_{b,l+1} = \hat{\mathbf{p}}_{b,l+1} + \Delta\mathbf{p}_{b,l+1},$$

with the optimal next sampling point's position $\hat{\mathbf{p}}_{b,l+1}$ and an offset $\Delta\mathbf{p}_{b,l+1}$, see Fig. 2.11. The term d_{sum} in (2.37) is split up into two parts as

$$d_{sum} = \underbrace{\sum_{i=1}^{l} dist(\mathbf{p}_i, \mathbf{p}_{i,s})}_{d_1} + \underbrace{\sum_{i=l+1}^{n} dist(\mathbf{p}_i, \mathbf{p}_{i,s})}_{d_2}.$$

For a strong task space bias the term d_1 is approximately equal for different branches

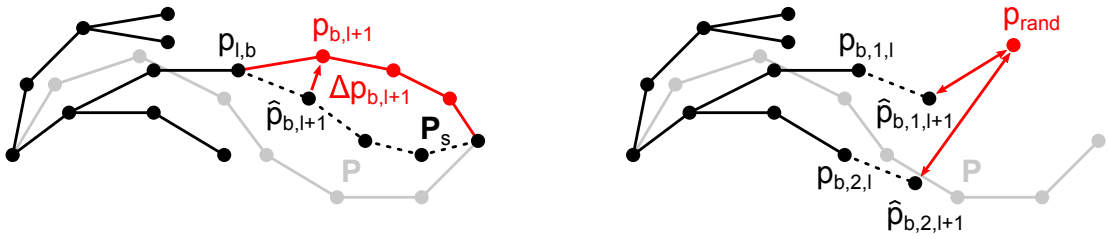


Figure 2.11: Task space nearest neighbors. Left: Sampling point position $\mathbf{p}_{b,l+1}$ being split up into an optimal position $\hat{\mathbf{p}}_{b,l+1}$ and an offset $\Delta\mathbf{p}_{b,l+1}$. Right: Resulting nearest neighbor search

of the tree. When expanding a tree branch by a new sampling point $\mathbf{p}_{b,l+1}$, the term d_2 depends quadratically on $\Delta\mathbf{p}_{b,l+1}$ while being minimal for $\Delta\mathbf{p}_{b,l+1} = \mathbf{0}$. It is thus sufficient to find the branch with minimal $\Delta\mathbf{p}_{b,l+1}$ when performing a nearest neighbor

search. Mathematically, when given different branches $\mathbf{P}_{b,j}$, finding the nearest sampling point $\mathbf{p}_{nearest}$ for a random sampling point $\mathbf{p}_{rand}$ reduces to

$$\mathbf{p}_{nearest} = \underset{\mathbf{p}_{b,j,l+1}}{\operatorname{argmin}} \|\mathbf{p}_{rand} - \hat{\mathbf{p}}_{b,j,l+1}\|_F.$$

An interesting observation is that the sensitivity $s_{l,n}$ for a trajectory with n sampling points, defined as

$$s_{l,n} = \left. \frac{\partial \left(\left\| \begin{bmatrix} \boldsymbol{\Omega}_1 \mathbf{L}_1 \\ \boldsymbol{\Omega}_2 \mathbf{L}_2 \end{bmatrix} \mathbf{P}_s - \begin{bmatrix} \boldsymbol{\Delta}_1 \\ \boldsymbol{\Delta}_2 \end{bmatrix} \right\|_F^2 \right)^2}{\partial^2 \Delta \mathbf{p}_{b,j,l}} \right|_{\Delta \mathbf{p}_{b,j,l}=0},$$

is largely independent of the choice of l and n and independent of the shape of the underlying path, see Fig. 2.12. This justifies the practice of using the nearest neighbor search regardless of the index of each node for the `Nearest` function.

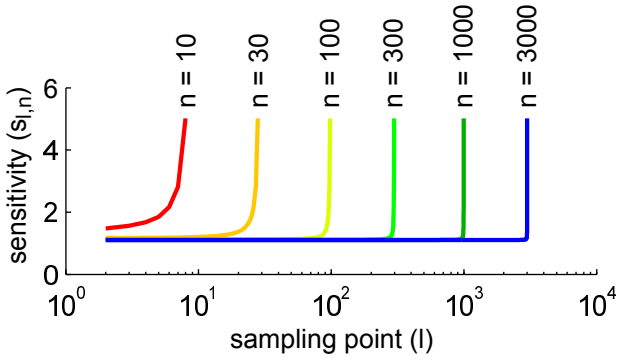


Figure 2.12: Sensitivity $s_{l,n}$ for multiple trajectories with different number of sampling points n, evaluated over varied sampling point indices l. The used parameterization is $\omega = 1e6$, $\omega_1 = 0.1$, $\omega_2 = 1$

Growing exploration ratio

Whereas both a task space bias and task space nearest neighbors are essential parts when combining LTE with the RRT* algorithm, there are further modifications possible which have proven to result in a considerable speedup of finding a valid solution. One is a growing exploration ratio. As displayed in (2.39) the `Approach` function depends on a variable α_r specifying the amount of exploration. When increasing its value with increasing iteration number n_r, the amount of exploration is increased in order to find trajectories which are less optimal but eventually bypass obstacles. This is expressed as

$$\alpha_r = \begin{cases} \phi_r \, n_r^{\kappa_r} & \text{if } \phi_r \, n_r^{\kappa_r} \leq 1, \\ 1 & \text{else,} \end{cases}$$

Multiple sampling point expansion

When expanding a branch of the tree, the optimal path $\mathbf{P}_s$ consists not only of the next sampling point, but all sampling points till the end of the path. Hence the branch $\mathbf{P}_b$ can expanded with multiple sampling points $\mathbf{p}_{b,l+1}, \ldots, \mathbf{p}_{b,l+\sigma_r}$ at every iteration without losing its optimality property. Fig. 2.13 illustrates the idea.

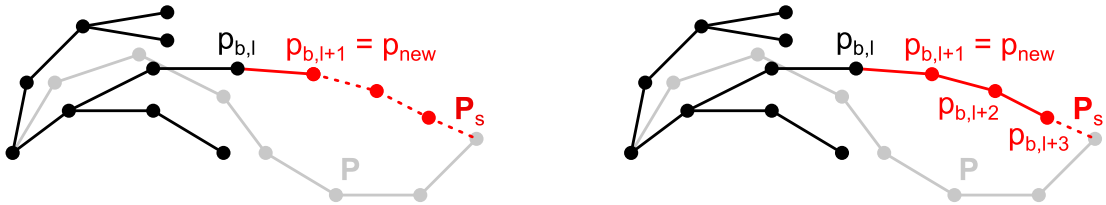


Figure 2.13: Multiple sampling point expansion. Left: Conventional approach extending only one node. Right: Accelerated approach improving multiple nodes ($\sigma = 3$ for the given illustration)

Reduced rewiring

For large search trees the `RewireId` checking every combination of $\{\mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i\}$ is the computational bottleneck of the method. Under the assumption of a tree consisting of n_t sampling points and all sampling point indices being identically distributed over the range $\{1, \ldots, n\}$, the number of checks per iteration is approximately

$$c_h = 3\left(\frac{n_t}{n}\right)^2,$$

resulting in an $O\left(\frac{n_t^2}{n^2}\right)$ complexity. In a similar manner to the task space nearest neighbor method the three sampling points are decomposed as

$$\mathbf{p}_q = \hat{\mathbf{p}}_q + \Delta\mathbf{p}_q, \quad q = \{i, i-1, i-2\},$$

into an optimal position $\hat{\mathbf{p}}_q$ and an offset $\Delta\mathbf{p}_q$. Then the total cost d_{sum} depends quadratically on $\Delta\mathbf{p}_q$. In this case the problem is reduced to find the node with minimal offset $\Delta\mathbf{p}_q$, which gives rise to the simplification of performing a nearest neighbor search over $\hat{\mathbf{p}}_q$ when rewiring the tree. In addition, complexity is reduced to $O\left(\frac{n_t}{n}\right)$ by rewiring only candidate sampling points with an index $id(y_{near}) = id_{rand} + 1$, see Alg. 6.

Spatial evaluation

A simulation experiment in 2D aims at displaying the principle procedure when expanding the Laplacian-RRT* tree. Shown in Fig. 2.14 are two different versions of the Laplacian-RRT* algorithm. For both versions the task consists in finding a trajectory that does not collide with obstacles (in red) while still resembling the reference trajectory (in

orange) as good as possible in terms of (2.38). The difference between the two versions is shown in Tab. 2.2. Task space bias and other related methods are only included in the biased version on the right side but not the unbiased version on the left side. Clearly the biased version resembles the original trajectory better than the unbiased one. Fig. 2.15

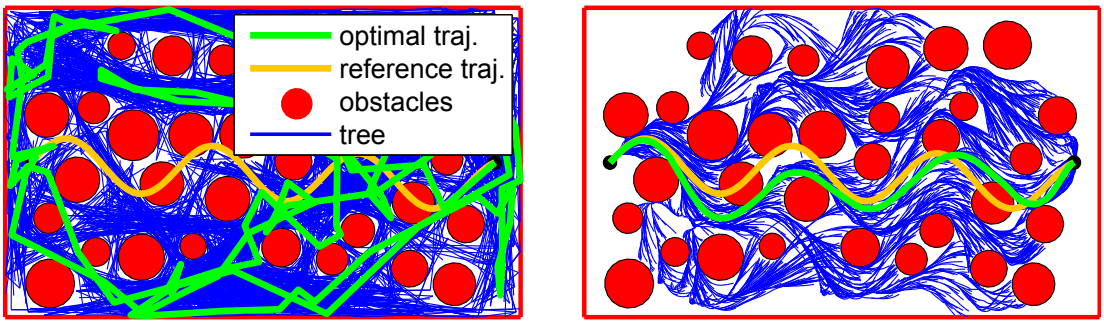


Figure 2.14: Spatial comparison of different Laplacian-RRT* approaches. Left: Unbiased version. Right: Biased version

method	unbiased version	biased version
number of iterations	$n_r = 5000$	$n_r = 10000$
weighting factors	$\omega = 10^6$ $\omega_1 = 0.1$ $\omega_2 = 1$	$\omega = 1e6$ $\omega_1 = 0.1$ $\omega_2 = 1$
task space bias	not used	used
task space nearest neighbors	not used	used
growing exploration ratio	not used	$\phi_r = 2e-4$ $\kappa_r = 0.5$
multiple sampling point expansion	not used	$\sigma_r = 3$

Table 2.2: Parameterization for the spatial comparison of different Laplacian-RRT* approaches

shows the mean cost d_{sum} of both unbiased and biased version based on 30 single roll-outs. Because each individual cost is only plotted after a valid solution has been found, different graphs start at different abscissa values. The cost of the biased version is about three magnitudes smaller than the cost of the unbiased version. Moreover, the graphs of the biased version are almost constant over a wide range of iterations indicating that due to the growing exploration ratio in combination with the task space bias, near-optimal solutions are found already at an early stage. The presented version of the Laplacian-RRT* algorithm minimizes the squared velocity and acceleration deviation from the reference trajectory. Fig. 2.16 shows velocity and acceleration plots of reference and optimal trajectory for the biased version. No major deviations from the reference trajectory are visible. The Laplacian-RRT* algorithm can be applied to arbitrarily shaped trajectories. In case of a straight trajectory, the optimal trajectory resembles the solution from the RRT* algorithm. Shown in Fig. 2.17 is the comparison between the RRT* and Laplacian-RRT*

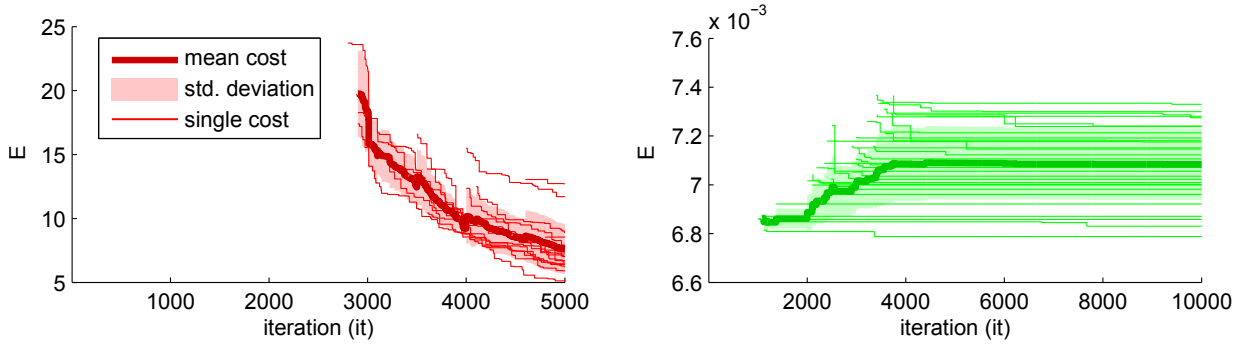


Figure 2.15: Cost comparison of different Laplacian-RRT* approaches. Left: Unbiased version. Right: Biased version

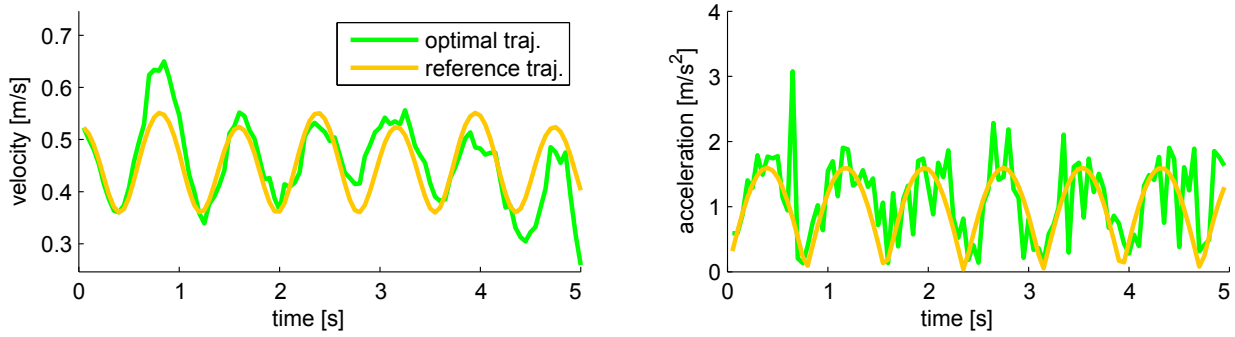


Figure 2.16: Velocity and acceleration plots of the optimal trajectory for the biased Laplacian-RRT* approach

algorithms. Although the optimal trajectories are almost similar, the created trees differ both in structure and shape due to different underlying principles of shortest path vs. minimal velocity/acceleration deviation. The task space nearest neighbor function relies on a strong task space bias. The stronger the bias is, the more the tree growth is shifted towards optimal solutions. The more optimal the solution is, the less rewiring operations of the `Parent(-Id)` and `Rewire(-Id)` function are necessary. Thus the amount of rewiring operations constitutes an indirect evidence for the strength of the task space bias. Shown in Fig. 2.18 is the relative amount of rewiring operations with respect to the number of new nodes $\mathbf{p}_{new}$ for the RRT*, the unbiased Laplacian-RRT* and the biased Laplacian-RRT* algorithm. Barely any rewiring is necessary using the biased Laplacian-RRT* algorithm, leading to the optional simplification of not performing any rewiring in order to reduce computation time.

Computational complexity

Shown in Fig. 2.19 are computation times averaged over 30 rollouts for a single-threaded Matlab R2013b implementation on a i7-3635QM/8GB notebook. Each gray bar indicates the overall computation time after every 1000-th iteration. It is split up both into al-

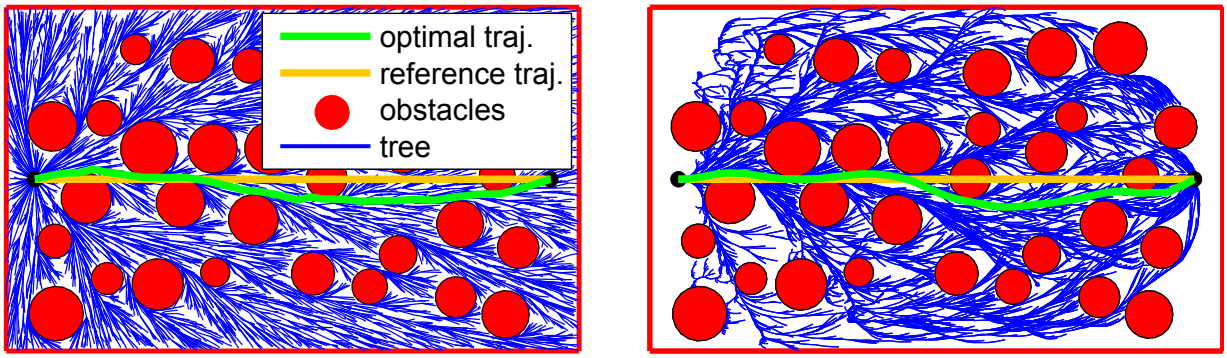


Figure 2.17: Spatial comparison of RRT* and Laplacian-RRT* for a straight reference trajectory. Left: RRT* approach. Right: Laplacian-RRT* approach

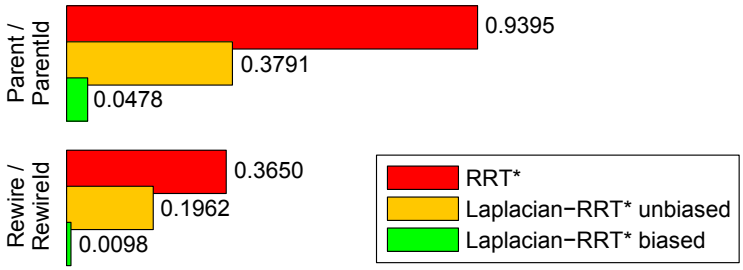


Figure 2.18: Percentage of rewiring operations for the `Parent(-Id)` and `Rewire(-Id)` function depending on the used algorithm

gorithms (`ParentId`/`RewireId`) and functions (LTE/collision detection/nearest neighbor search) causing the main computational load. In addition, the number of tree nodes and its standard deviation is displayed in blue. It is visible that most computation time is spent for the nearest neighbor search and collision detection. Whereas the method has a theoretical linear complexity when using optimized nearest neighbor search method, Matlab's built-in `knnsearch` function causes the given example to scale quadratically with the number of tree nodes.

2.10 Experimental evaluation

This section focuses on real life experiments, verifying the capabilities of LTE and showing how the presented approaches and expansions can fulfilling a variety of typical robotic tasks.

2.10.1 Volleyball scenario with a planar 3DOF robot

A first experiment aims at displaying the online deformation capabilities of LTE while sufficing additional boundary constraints in a dynamic manipulation task. Inspired by the game of volleyball, a tilted airhockey table and a round puck are used to construct a 2D

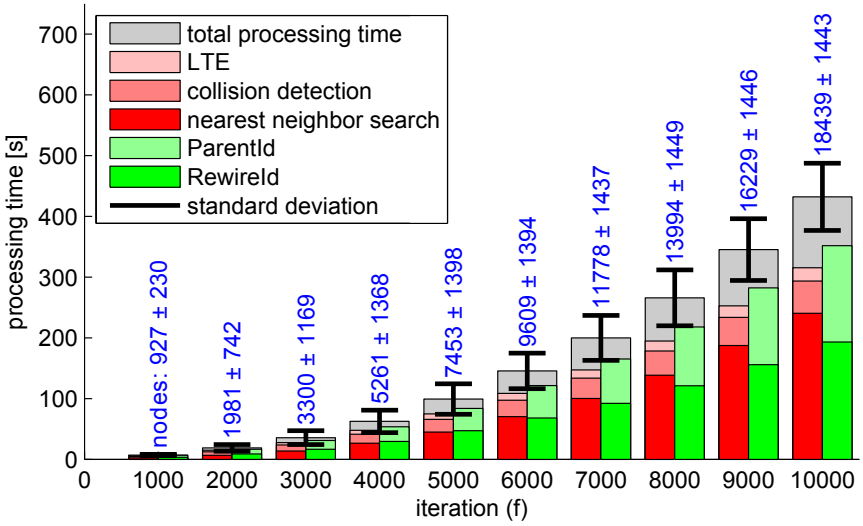


Figure 2.19: Processing time evaluation of the Laplacian-RRT* approach

representation of the game. When playing, a human player faces a robot opponent. The goal is to hit the puck such that it moves across an obstacle (net) located in the center of the table and lands in the opponent's field. The robot player consists of a planar 3DOF manipulator with a flat EE plate. For an incoming puck the system automatically calculates both, an optimal outgoing puck trajectory and the necessary hit position/velocity of the EE, see App. A. Five infrared markers make it possible to track the position of the puck by a Qualisys motion capture system operating at $250Hz$ while the robot operates with a sampling rate of $1000Hz$. Both motion planning and motion controller are implemented in Matlab/Simulink and compiled with Mathwork's Realtime Workshop. The airhockey table measures $160 \times 80cm$, out of which only the right half is accessible by the robot. Whereas the robot's maximum speed is $7\frac{rad}{s}$ per joint, the underlying PD position controller constitutes the limiting factor. An reference minimum-torque trajectory for a typical hitting motion is calculated offline using DirCol [307, 242]. As the method is to slow for online replanning, LTE is used to deform the reference trajectory with minimum acceleration deviation. Constraints limiting the amount of deformation are given by the boundaries of the airhockey table and the net in the middle of the table. A spline-based deformation in combination with high-weighted positional constraints and convex hulls for collision avoidance as described in Sec. 2.6.2 is used to suffice all constraints and replan a new trajectory within one sampling step. To derive tighter spatial bounds of the convex hulls during trajectory deformation, the trajectory is split up into smaller trajectory segments of about $50ms$ length. After throwing the puck, the system automatically estimates the trajectory of the puck through LS regression based on 20 frames of the motion capture system, corresponding to a $80ms$ delay until the desired EE trajectory is calculated. Execution of the desired EE trajectory begins immediately after its calculation, requiring an adaptation of the optimal trajectory calculated with DirCol not only in the spatial but also temporal domain. A typical batting motion lasts around $800ms$, consisting of a $350ms$ long interval from the rest position to the hit position and a $450ms$

long interval where the robot returns back to its rest position. Fig. 2.20 shows an example in which the puck is hit by the human and batted back by the robot. The online

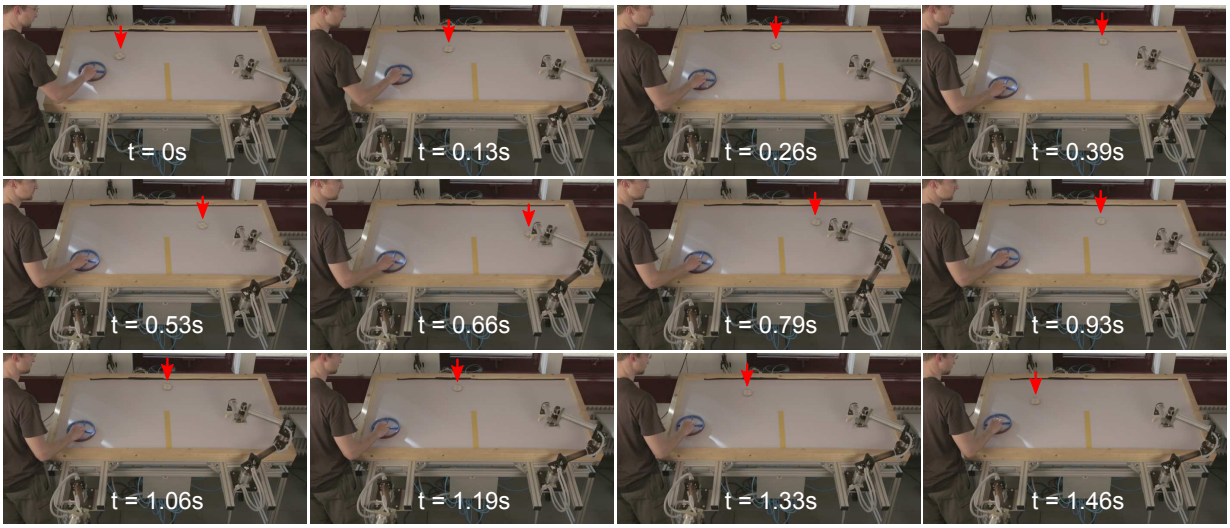


Figure 2.20: Volleyball scenario with a human playing against the robot. A red arrow highlights the position of the puck

trajectory deformation is illustrated more in detail in Fig. 2.21. The left side shows a variety of possible trajectory deformations depending on the hit point of the puck (blue dots) and the resting start/end position of the trajectory (red dot). For the given example the desired velocity in the moment of hitting is similar for all hit points. The color of the path corresponds to the EE velocity during movement, with green indicating low velocities and red for high velocities as indicated by the color bar on the right side. Because the time to reach the hit point and the time back to the resting position are similar for all trajectories (381*ms* and 673*ms*), one can see well how the EE has to move faster to reach a hit position further away from the resting position. The blue line marks the undeformed reference trajectory calculated with DirCol. On the right side a detailed view of the obstacle avoidance is shown. To avoid the workspace boundaries (red line), convex hulls (red rectangles) based on the spline representation of the deformation process and the undeformed trajectory are calculated as shown in (2.28). As they enclose the trajectory, the trajectory is deformed through the collision avoidance scheme of Sec. 2.9.1 such that the convex hulls do not intersect with any obstacle (green rectangles). The principal goal of torque-optimal trajectories is only indirectly tackled by the deformation using splines. Even if the approach minimizes the acceleration difference, the absolute acceleration can still become large for big trajectory deformations. As the EE acceleration in task space is coupled to the manipulator torques through the dynamical model of the manipulator, minimizing acceleration deviation also minimizes torque deviation to a certain extent. The faster the movement is, the more dominant the effect becomes as gravitational terms play only a minor role in this case. Fig. 2.22 illustrates the effect. The approach works reliably in practice and successfully avoids workspace constraints. The combination of a spline-based approach with a convex hull-based collision avoid-

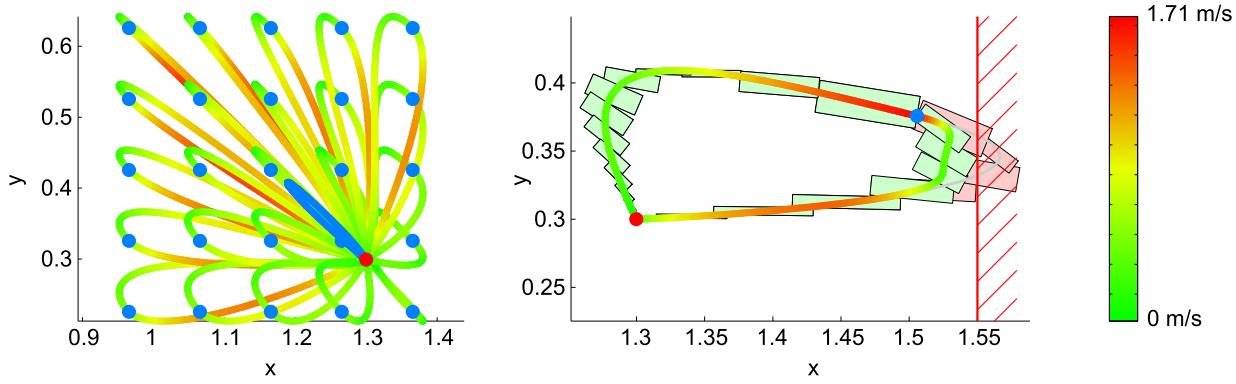


Figure 2.21: Trajectory deformation for the volleyball scenario. Left: Multiple possible deformations for a given hit position (blue dots) and start/end position (red dot) with corresponding velocity along the trajectory (path color). Right: Collision avoidance scheme using convex hulls for a given workspace boundary (red line)

ance strategy is the fastest LTE adaptation method known so far. For the given example the convex hulls of both the original trajectory segment and the additive LTE term consists of only four sampling points, resulting in 16 sampling points after calculating the Minkowski sum. Compared to 50 sampling points for each trajectory segment of $50ms$ length computational complexity for collision avoidance is reduced by a factor of 3. Although not implemented in the current experiment, an optimal manipulator trajectory can be recalculated every millisecond for a fast adaptation in case of sudden disturbances or nonlinear friction effects that are not considered in the actual model. The approach is very stable as it only depends on a few tunable parameters. Among these the length of each trajectory segment ($50ms$ for the given example) is the most crucial one as it constitutes a tradeoff between two extrema. When chosen too high, computational complexity is further reduced for collision avoidance but the resulting convex hull from the Minkowski sum may be too conservative. When chosen too small ($1ms$ in the limiting case) the advantage of using convex hulls is canceled out.

2.10.2 Bin-the-litter scenario using a HRP-4 humanoid robot

The second scenario displays how LTE can be combined with a prioritized inverse kinematics (IK) approach to generate whole-body motions for a humanoid robot. It also shows an application making use of low-weighted positional constraints for collision avoidance and the cooperative manipulation scheme to carry an object with two hands. The task is to clean up garbage and consists of lifting a bin from a lower position onto the table, grasp a bag of garbage firmly with both hands and dispose the garbage in the bin. The human demonstration is recorded at a frame rate of $200Hz$ using a motion capture system from Motion Analysis Inc. by tracking the position and orientation of both hands, bin and the garbage bag. The obtained motion is then mapped to a HRP-4 robotic platform and adapted using LTE. The used HRP-4 platform is about $151cm$ tall and has 50 joints,

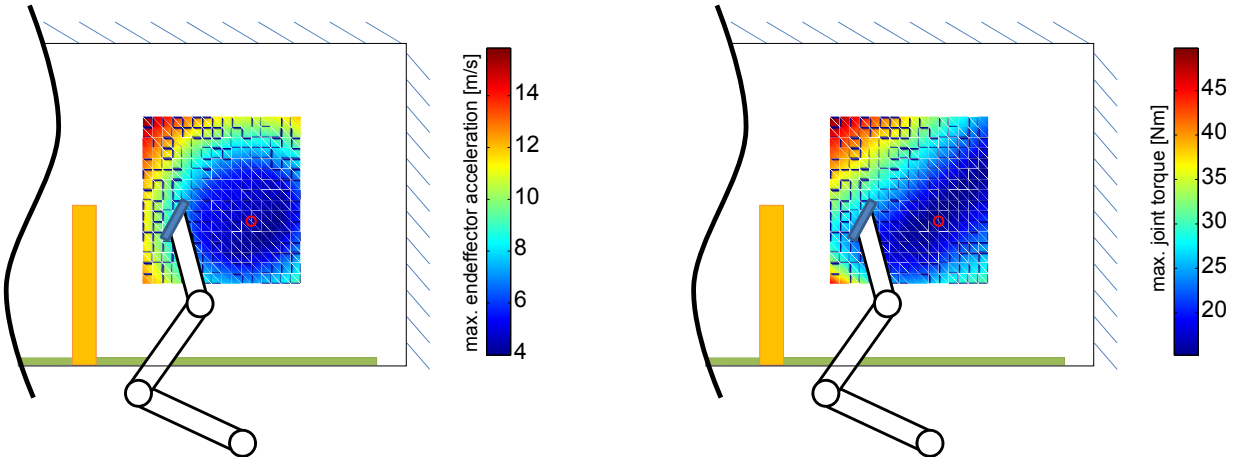


Figure 2.22: Maximum EE acceleration and maximum joint torque depending on the hit position in task space. A red dot indicates the hit position of the reference trajectory

out of which 34 are articulated and used for task reproduction [139]. It weights about $39kg$ and is controlled by a $1.6GHz$ Intel Pentium M prozessor. First simulations are conducted in Matlab to synthesize the desired hand pose for a successful task completion. Based on the result, a whole-body motion is calculated in OpenRave [69] and verified afterward in OpenRTM [12] before final execution on the robot. A prioritized IK approach ensures a physically consistent whole-body motion taking joint angle limitations, self-collisions and center of mass (COM) based balance into account for a given trajectory of both hands. The multiresolution approach of Sec. 2.7 is used to fit the motion of the human demonstration to the different figure of the robot, resulting in the imitation run in Fig. 2.23. All LTE parameters are displayed in Tab. 2.3. The modification run differs from the adaptation run in two ways: An added obstacle (yellow book) is placed on the table and must be avoided by the robot when putting the bucket on the table. In addition the initial position of the garbage bag is elevated by around $40cm$. By creating a repellent force field around the book and using low-weighted positional constraints as described in Sec. 2.9.2, the obstacle is circumnavigated. To account for the modified position of the garbage bag, the hand trajectory is lifted accordingly using positional constraints. During both runs the cooperative manipulation scheme of Sec. 2.8 maintains a specific distance between both hands when holding the garbage bag, preventing it from falling down and grasping the bag firmly. For a better understanding of the movement adaptation process, the trajectories of both EEs are compared in Fig. 2.24 for the imitation and adaptation run. Based on the marker position attached to bin, garbage bag and the table, their position is reconstructed. Whereas the table is shown in gray, both garbage bag and bin before and after the placement are marked with blue. When comparing the imitation run on the left side with the adaptation run on the right, it is clearly visible how the trajectory of the left endeffector is lifted to avoid the obstacle. Fig. 2.25 shows velocity and acceleration plots for both EEs during the imitation and adaptation run after LTE deformation. Whereas both human demonstration and robotic motion last around

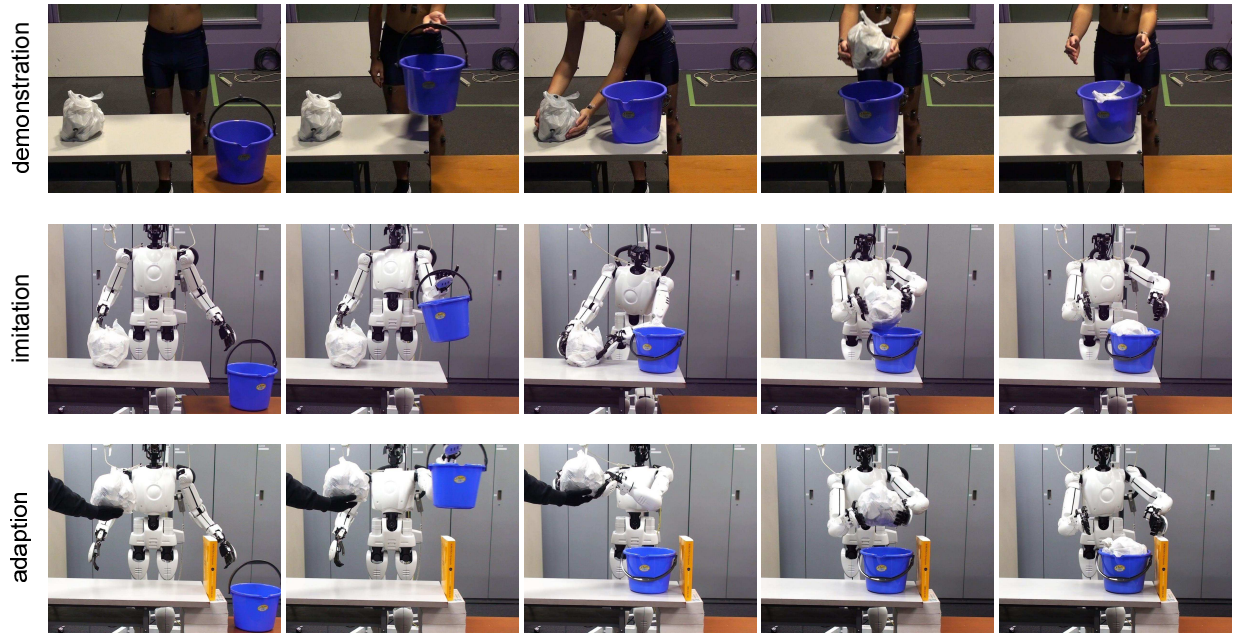


Figure 2.23: Bin-the-litter scenario. Top: Human demonstration. Middle: Robot imitation. Bottom: Robot adaptation

7.5s, the motion is slowed down afterward by a factor 5 for safety reasons. Both velocity and acceleration are continuous without any visible jumps. It is good to see how velocity and acceleration of the left EE increase during the adaptation run as the robot has to circumnavigate the object without retargeting the time. When holding the garbage bag, the original distance between both EEs during the imitation run is maintained also during the adaptation run. This effect is displayed in Fig. 2.26. To avoid a collision with the added obstacle, the path of the EE is modified to a large extent for $2s < t < 3s$, thus the distance between the imitation and adaptation run differs. For $6s < t < 7s$ the robot holds the garbage bag with both endeffectors, making a precise coordinated movement essential. Applying the cooperative manipulation scheme lets the distance and also the relative spatial position between both EEs stay identical for the adaptation

parameter	value
α_w	0.001
β_w	0.001
γ_w	3
ϵ_w	0
ϕ_w	0.001
ζ_w	0
ω_c	$1e4$
ω	$1e6$

Table 2.3: Parameters for the bin-the-litter scenario

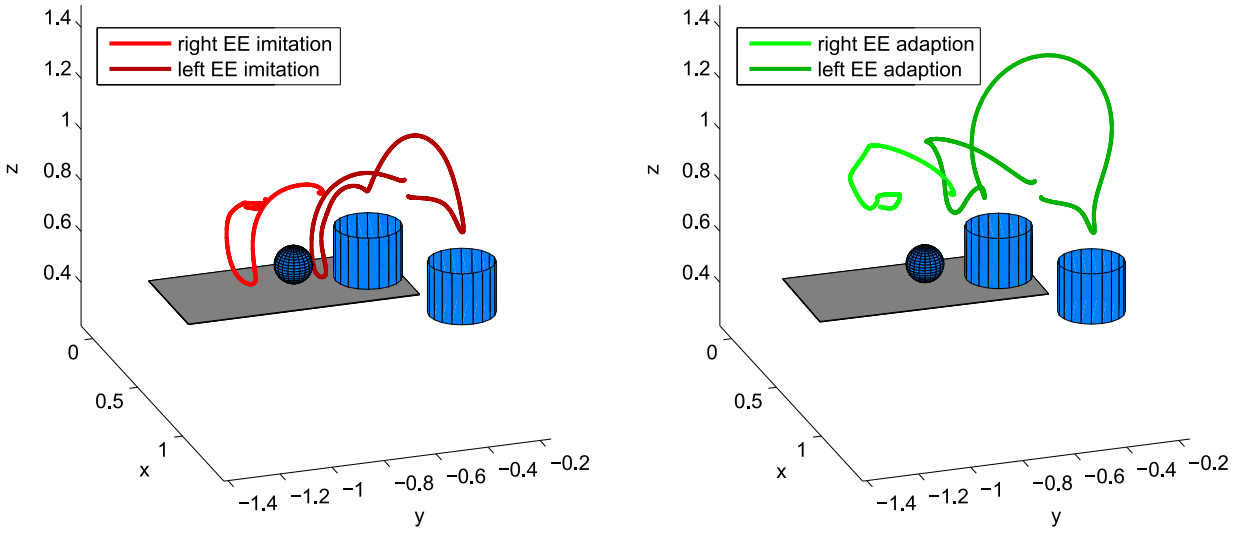


Figure 2.24: Bin-the-litter scenario. Left: Schematic view of the imitation run. Right: Schematic view of the adaptation run

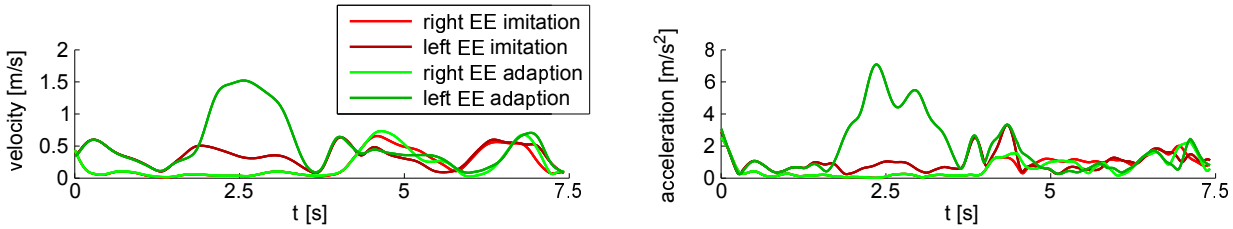


Figure 2.25: Bin-the-litter scenario. EE velocity and acceleration for both hands during the imitation and adaptation run

and imitation run. The example shows how a complex motion can be adapted in a goal-directed way. Compared to a probabilistic encoding using HMMs or a vector-field based representation, the trajectory-based approach allows an intuitive understanding of the deformation process and eases verification. Still, as only the EE trajectories are adapted through LTE it neglects the underlying robot kinematics. Although the robot is highly redundant, the constraints beside LTE limit the workspace considerably, thus allowing only small amounts of adaptation. The low-weighted positional constraints depends on a few parameters and required some fine tuning for motion adaptation. Here a tradeoff had to be found between a large safety distance to any obstacle and the limited workspace of the robot, requiring close circumnavigation around obstacles.

2.10.3 Lift-the-cup scenario using a HRP-4 humanoid robot

A final scenario illustrates the planning capabilities of the Laplacian-RRT* algorithm in Sec. 2.9.3 in a real environment. The task consists of lifting a cup from under the table onto the table. Similar to the previous example a reference motion recorded from a

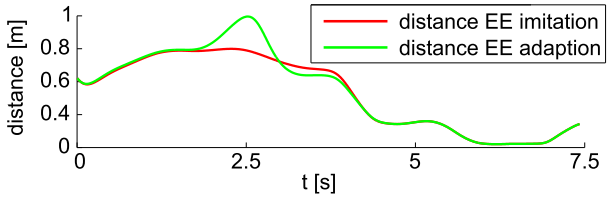


Figure 2.26: Bin-the-litter scenario. EE distance between both hands during the imitation and adaptation run

human demonstration is mapped to the robot using a motion capture system from Motion Analysis Inc. and mapped to the robot. Due to a stack of books being placed on the table a pure motion imitation leads to an unsuccessful task completion as the robot collides with it. The Laplacian-RRT* method is used to find an optimal, non-colliding motion resembling the reference motion as good as possible. The 3D search space consists of the position of the right hand, with its orientation kept similar to the reference motion. A prioritized IK approach ensures that additional constraints like joint angle limitations, self-collisions and COM-based balance are maintained. If the constraints are violated or the robot collides with an obstacle, the corresponding tree segment is discarded when expanding the tree. Results are shown in Fig. 2.27 with the used parameters given in Tab. 2.4. Used mainly for collision avoidance, the approach is also compared with a conventional potential fields approach of the form

$$V(\mathbf{p}) = \begin{cases} \frac{1}{2}k_r(d_0 - d(\mathbf{p}))^2 & \text{if } d(\mathbf{p}) < d_0, \\ 0 & \text{otherwise.} \end{cases}$$

for a given minimal point-obstacle distance $d(\mathbf{p})$ and scalar constant d_0. As shown in the figure, the similarity measure d_{sum} for the Laplacian-RRT*-based trajectory is about 5× better (=lower) than for the potential field based trajectory. The entire robot motion is first simulated in OpenRave [68] and verified in OpenRTM [12] before execution.

parameter	value
n_r	1000
ω	$1e6$
ω_1	0.1
ω_2	1
ϕ_r	$1e-4$
κ_r	0.6
σ_r	10

Table 2.4: Parameters for the lift-the-cup scenario

Shown in Fig. 2.28 are velocity and acceleration plots for the optimal trajectory found by Laplacian-RRT*, the potential field based trajectory and the reference trajectory. Whereas the reference trajectory is almost perfectly tracked for $t < 2.5s$ using potential fields, the

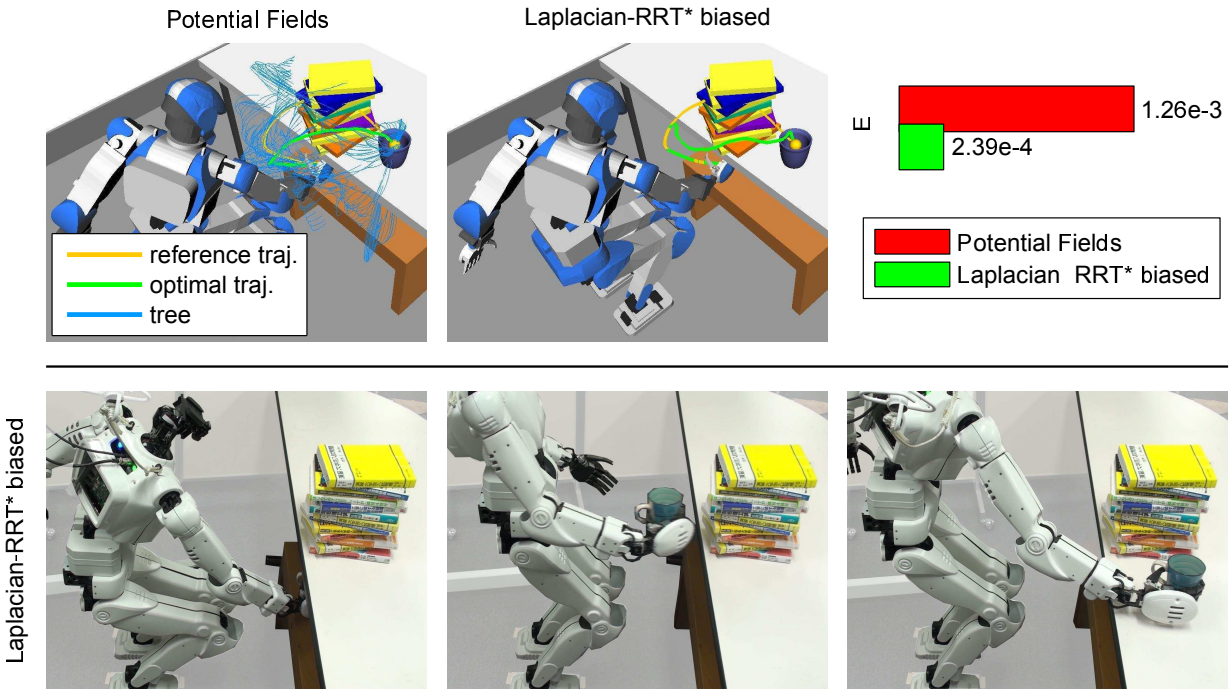


Figure 2.27: Lift-the-cup scenario. Top: Simulation results. Bottom: Conducted experiment. A bar graph compares the similarity measure d_{sum} for the Laplacian-RRT* and an artificial potential field approach

repellent force fields induced by the obstacles cause large velocity/acceleration deviations for $t > 2.5s$. The Laplacian-RRT* trajectory has only a small velocity/acceleration error over the entire time span. The small ripples are caused by the stochastic nature of the algorithm and become smaller with increasing number of sampling point of the Laplacian-RRT* search tree. Yet the potential fields approach allows for online adaptation whereas the Laplacian-RRT* approach must be calculated offline. The presented approach shows

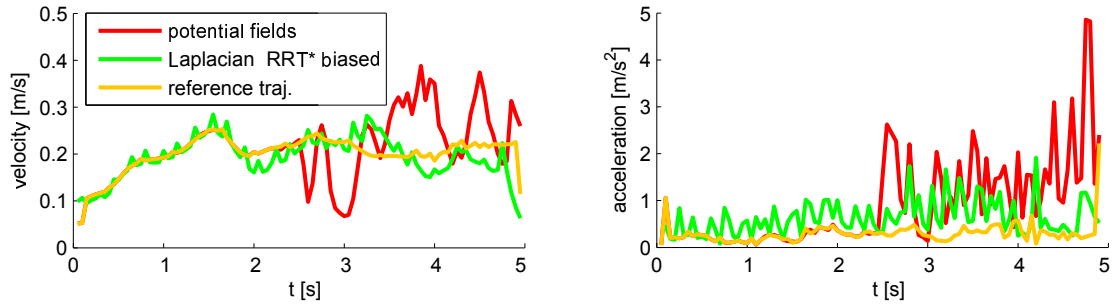


Figure 2.28: Lift-the-cup scenario. EE velocity and acceleration of the right hand for the potential fields approach, the reference trajectory and using Laplacian-RRT*

how the Laplacian-RRT* algorithm can be used to imitate a given motion in a constrained environment. Yet its general nature does not limit application to kinematic constraints for a fixed robot. It also allows to consider more complex aspects like dynamics-based

constraints, advanced scenarios where the robot also has to walk or the inclusion of an underlying controller. If any of the conditions is violated, it is interpreted as the collision of the tree segment with an obstacle and the tree segment is discarded. However, as both the original RRT* and the Laplacian-RRT* algorithm depend on a fast collision checking routine such approaches are limited to robotic systems with few DOF [312, 130, 143, 144].

2.11 Conclusion

This chapter introduces LTE as a powerful tool for trajectory and path adaptation. The method's key aspect is to represent any constraint during path adaptation by a set of linear equations which can be included in the LTE framework and solved efficiently using LS. Whereas the original formulation is of general nature and only of limited scope for complex motion adaptation, several modifications and expansions fit the approach to robotic specific needs like passing waypoints or avoiding obstacle while maintaining a smooth trajectory. Special focus is drawn on discussing both theoretical properties that might be of interest for future expansions and practical aspects engineers have to consider during implementation. As such a large part of the chapter is dedicated to computational complexity and possible improvements to increase the speed of the method. The method requires a discretized reference trajectory for adaptation, thus being unusable for other trajectory descriptors like state-space representations or a probabilistic encoding. In addition the LS approach provides only an approximate solution for overdetermined equation system. Whereas this is crucial from a theoretical point of view as constraints are never perfectly met, the practical implications are minor as the error can be made insignificantly small before encountering numerical instabilities.

2.12 Bibliographical notes

The results in this chapter and the related appendix are either partly or in their entity based on the author's work in [223, 225, 226, 228]. Because LTE is based upon the idea of interpreting a discretized trajectory as a surface mesh, the original approach has strong similarities with existing literature in computer graphics on mesh deformation, for example the handling of nonlinear deformation effects for surface meshes with arbitrary node connectivity [280]. The presented approach in this chapter is only a special version of Laplacian mesh processing, allowing simplifications due to the unique mesh structure of paths. In a similar fashion there are multiple articles on multiresolution approaches downsampling a mesh through subdivision [157, 159], yet they are specifically designed for surface meshes and thus not directly applicable to paths. When calculating spatial bounds of LTE, both the concept of a leader-follower network and the decomposition of the system dynamics matrix into multiple submatrices are common in networked controlled systems [198] and only adapted in this chapter to modify paths. For collision avoidance, the LTE framework is combined both with a potential fields approach and with the RRT* algorithm, Both methods are independent concepts presented also in

[141, 151] and modified to cirrcumnavigate obstacles while maintaining the shape of the rreferrence.

3

Motion imitation and control

This chapter investigates control structures to generate robotic motions which share the common optimality principles as the corresponding motion adaptation scheme. Differing from existing state-of-the-art methods the approach allows to consider not just kinematic constraints but also dynamic constraints while maintaining local trajectory properties by means of a predictive optimal control scheme.

The chapter is organized as follows: Focusing on whole body motion control in Sec. 3.4, a classical prioritized IK scheme is combined with a LQR controller to account for LTE-specific local trajectory properties. It also presents an HMM-based probabilistic approach considering not only the movement of the robot but also its interactions with the environment. For EE control subject to trajectory retargeting a piecewise linear system based approach is presented in Sec. 3.7. Focusing on dynamic manipulator properties in Sec. 3.5, a closed-form algebraic expression is derived to specify upper bounds on the amount of deformation that suffice given torque constraints. An trajectory classifier is presented in Sec. 3.8 that shares common properties with LTE bus is applicable to a wider class of classification problems. Most proposed control and classification schemes are evaluated in Sec. 3.9 through experiments. Sec. 3.10 provides a short summary about the chapter. The chapter ends with Sec. 3.11 where bibliographical remarks to other published articles about the topic are listed.

3.1 Introduction

LTE as presented in the last chapter is used to plan and adapt a desired movement before execution. This happens on an abstract trajectory level which is at first independent of the used manipulator structure and takes a perfect environment for granted, e.g. no dynamic

constraints or disturbances caused by external influences or measurement inaccuracies. In contrast, this chapter focuses on the question how a predefined motion is executed best on a given robotic platform. It comprises of different aspects that must be taken care of when being applied to the field of motion imitation.

The first aspect is motion representation. For a calculated motion, the type of motion encoding influences the imitation capabilities and is crucial for successful task completion. This aspect also considers the data to use when specifying the task as a brute-force approach considering all available data might lead to no additional insights if both the task-specific and imitation-related properties must be extracted and the resulting information are contradicting. As for most tasks rather few conditions must be met, it gives room for modifications of the original motion which do not interfere with the task. Still these modifications must be optimized by measuring their influence on the imitation quality during execution.

The second aspect is motion mapping. If the motion is transferred from a demonstrator to a robot with different kinematic and dynamic properties, additional conditions must be met by the motion imitation scheme. A simple one-to-one mapping is in general impossible to be executed by the robot as violated joint angle limits will damage gears, additional balance constraints may cause the robot to fall over or different link lengths can cause positioning errors.

A last aspect are additional considerations of motion control in the context of motion imitation and adaptation. Differing from a conventional, decoupled approach between planning and control this thesis favors a tight coupling between the motion adaptation methods in the last chapter and the motion control methods in this chapter. The underlying idea is to find control schemes that share the same optimality principle as the motion adaptation schemes, thus easing the design process and avoiding contradicting control schemes.

3.2 Related work

As LTE is primarily designed to deform EE trajectories in task space, it requires a mapping scheme from task to joint space for a feasible whole-body motion. Different methods and applications exist for the IK problem [107, 129, 269], with differential IK methods as the most common ones [217]. Although the associated Jacobian matrix is in general calculated analytically, recent advances in the field of (un-)supervised learning make it possible to learn the IK [75, 275]. The differential IK approach accounts for arbitrary task spaces and is not limited to EE trajectories. There are different ways to define such task spaces, for example through a probabilistic encoding scheme using HMMs [170] that accounts for temporal deviations or through automatic extraction [131, 132, 213]. They can be also defined through offline comparison [169] or encoded in a Dynamic Bayesian Networks to deal with unforeseen perturbations [80], making clear that the used task space cannot be uniquely defined and depends highly on the user requirements.

Every IK scheme relies on a controller to suppress disturbances and let the manipulator converge back to its desired position in case of an sudden offset. To be consistent with LTE, a controller must consider the same optimality principle. Suitable control methods

considering the time-varying structure of the local trajectory properties are found in the field of optimal control [203] and model predictive control [88]. Both control methods are closely related. Optimal control aims at finding a control law for a dynamical system such that a given optimality criterion is met. Model predictive control has the same goal but is solely used for discrete models and a finite control horizon, whereas optimal control also considers the continuous case and an infinite control horizon. An important class of optimal controllers is the linear quadratic regulator or LQR controller for linear systems and quadratic cost functions [174]. Multiple extensions exist, including applications to nonlinear systems [191] and stochastic/risk-sensitive systems [200, 267], global optimal control when combining LQR with RRT* [100, 295], adaptive LQG controllers for collision avoidance [28] and constrained problems [264]. Generic and powerful the LQR controller is, it is also computationally demanding. A simpler control version in this chapter is based upon a set of piecewise linear (PL) systems [277, 133] which are quickly recalculated whenever the reference trajectory gets deformed. Extensive research is conducted on PL systems regarding the synthesis to stabilize and control nonlinear systems and their identification [87]. The approach is driven by the idea to create a vector field that automatically avoids obstacles and converges to a desired goal state [103, 149, 155, 182, 193, 235].

If not only kinematic but also dynamic constraints have to be considered, a precise model of the robot is required to simulate the inverse dynamics [84, 220]. Once obtained, it can be used to suffice dynamic constraints by either retargeting the motion in the temporal or spatial domain [35, 245]. Both types of approaches can also consider other dynamics-related effects like unilateral constraints and rigid contacts [256].

Once a desired motion is executed on a robot by means of an underlying control scheme, it's similarity to a given reference motion must be evaluated and classified in the field of motion imitation. LTE is very limited in this context as the trajectories to be classified must fulfill restrictive constraints. There exist more generic trajectory-based descriptors which are categorized according to the invariant features they consider [45, 18, 71, 42]. Without any invariant features, trajectories must match both in the spatial and temporal domain for being classified as similar. Temporal invariance allows two trajectories to differ in temporal domain while following the same path for being considered as similar, see for example dynamic time warping (DTW) [148]. Regarding spatial invariance there exist multiple descriptors, based either upon force fields [60], pose normalization through principal component analysis (PCA) [22, 209] or curvature/torsion along the trajectory [65, 317]. An additional distinction is made between non-hierarchical descriptors working on a certain granularity level of the trajectory like LTE [306] and hierarchical descriptors capturing both local and global trajectory properties [41, 186, 316]. Depending on the used classifier, a motion is encoded not through a temporal-spatial trajectory [234] but a symbolic representation [171, 201] that is used to generate full body movements [24, 135, 291] and has in general better abstraction capabilities than a trajectory-driven approach.

3.3 Problem statement and conceptual approach

Regarding the original problem in (1.1), which is stated again as follows

$$\begin{aligned} &\text{optimize } g(x, \hat{x}), \\ &\text{s.t. } h(u), \end{aligned}$$

the goal of motion adaptation was to optimize g such that the adapted motion x matches the original motion $\hat{x}$ best while sufficing additional constraints in $h(u)$. Motion control achieves the same goal but focuses on online control instead of offline planning. Whereas the latter one is necessary to find an initial solution for an underlying control scheme, it considers an ideal environment. However during execution an online controller has to deal with all hardware-related challenges like measurement inaccuracies or external disturbances. In addition there are hard real time constraints imposed on an embedded controller as a new optimal input must be calculated at every sampling step. Offline planning methods do not suffer from these constraints.

The major challenge of motion control in the field of motion imitation is consistency with the motion planning framework in order to minimize a similar cost function. Classical manipulator control schemes are inapplicable as they focus solely on convergence of the resulting trajectory but not its local properties. Optimal controllers on the other hand are more flexible as they can be based on an arbitrary cost function, yet are difficult to prove for convergence and computationally demanding. Existing schemes like GMM and DMP guarantee convergence but lack the optimality property in case of a disturbance.

Thus the main contribution of this chapter is to derive a controller consistent with LTE that can be executed in real-time and has guaranteed convergence properties. To account also for whole-body control, LTE is combined with a prioritized IK approach. As LTE considers the local trajectory properties along the entire trajectory, these information must be also processed by the controller. When linearizing the IK around a certain operating point, the optimal input is calculated through an LQR controller while considering local trajectory properties over a finite horizon. If the trajectory has to be modified online, a hybrid control/planning framework is presented which allows trajectory deformations at every time step while ensuring asymptotic stability towards a reference trajectory. Despite being powerful, the method is also computationally expensive as the input has to be optimized over a number of future steps via backward induction. A simpler scheme based on piecewise linear systems overcomes this drawback. By creating a set of convergent vector fields around the reference trajectory which are recalculated whenever the trajectory gets deformed, stability is guaranteed within specific bounds. To reduce the number of calculations during online motion control while providing a generic descriptor for motion imitation and adaptation tasks, a probabilistic learning-based scheme is presented. By encoding the task space distances between robot links and objects in the environment, it automatically extracts whole body pose information of the robot and interactions with other objects. Focus is drawn on suitable simplifications for preprocessing most information before execution.

3.4 Whole-body control and imitation

This section presents control schemes which share the same optimality principle as LTE and expand it to whole-body movements. As the main purpose of LTE is to plan desired trajectories, it requires an underlying controller when being executed on a real robot. In case of a sudden disturbance causing the executed trajectory to differ from the reference trajectory, LTE alone neither accounts for collision avoidance of the executed trajectory nor for convergence back to the reference trajectory. Moreover it relies on a IK scheme if a robot with multiple DOFs is considered.

The section presents three approaches. The first one is a conventional differential prioritized IK scheme for a Jacobian-based mapping of a task space motion to joint space. Based on the differential IK representation a LQR-based controller is developed, tracking a reference trajectory in a predictive manner while avoiding singularities through an adaptive weighting scheme of task space and joint space objectives. Both approaches can be combined with LTE by augmenting either the state vector of the resulting dynamical system or the cost function, resulting in a novel LQR controller for continuous trajectory retargeting. The last approach works not a single EE trajectory but instead defines a hyperdimensional task space trajectory by considering a set of spatial task space distances between links of the robot and objects in the environment, automatically extracting the task-relevant features and layering the foundation for a subsequent differential kinematics imitation scheme.

3.4.1 Prioritized inverse kinematics and singularity avoidance

This section presents a common prioritized IK scheme and shows how it can be combined with LTE for a continuous replanning of the desired trajectory. It is assumed that the task space coordinates $\mathbf{x}$ are related to the generalized joint space coordinates $\mathbf{q}$ through the differential relation

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \tag{3.1}$$

with $\mathbf{J}$ as the Jacobian matrix encoding the differential kinematic relation between joint and task space. If both vectors $\mathbf{x}$ and $\mathbf{q}$ are of similar rank, an inverse solution is calculated by means of the inverse Jacobian as

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}\dot{\mathbf{x}},$$

In all other cases, the Moore-Penrose pseudoinverse is used to find the solution minimizing $\dot{\mathbf{q}}^T\dot{\mathbf{q}}$ for a given task space velocity $\dot{\mathbf{x}}$ as

$$\begin{aligned}\dot{\mathbf{q}} &= \mathbf{J}^{+}\dot{\mathbf{x}} \\ &= (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\dot{\mathbf{x}}.\end{aligned} \tag{3.2}$$

By using the right pseudoinverse it is implicitly assumed that $\text{rank}(\mathbf{q}) \geq \text{rank}(\mathbf{x})$. The approach fails for kinematic singular configurations as in this case the term $(\mathbf{J}^T\mathbf{J})^{-1}$ is not invertible anymore. The problem can be overcome by adding a small constant term

$\pi_d \mathbf{I}$, $\pi_d > 0$ when calculating the inverse. This is known as dampled least squares (DLS) and results in

$$\dot{\mathbf{q}} = (\mathbf{J}^T\mathbf{J} + \pi_d \mathbf{I})^{-1}\mathbf{J}^T\dot{\mathbf{x}}. \tag{3.3}$$

Differing from (3.2), the inverse in (3.3) can always be calculated but results in a nonzero tracking error $\|\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}\| > 0$. To overcome this problem, the singular robust (SR) inverse or WDLS is introduced in [217] as

$$\begin{aligned}\dot{\mathbf{q}} &= \mathbf{J}^{\#}\dot{\mathbf{x}} \\ &= (\mathbf{J}^T\mathbf{W}_x\mathbf{J} + \mathbf{W}_q)^{-1}\mathbf{J}^T\mathbf{W}_x\dot{\mathbf{x}}.\end{aligned} \tag{3.4}$$

with $\mathbf{W}_x$ and $\mathbf{W}_q$ as positive definite weighting matrices related to the errors in task and joint space. For a parameterization of $\mathbf{W}_x = \mathbf{I}$ and $\mathbf{W}_q = \pi_d \mathbf{I}$ the solution equals the DLS solution, for $\mathbf{W}_x = \mathbf{I}$ and $\mathbf{W}_q = \mathbf{0}$ it results in the Moore-Penrose pseudoinverse solution. The key idea of the weighted damped pseudoinverse is to adjust the elements of the weighting matrices dynamically. Whereas for $\mathbf{W}_x$ a common parameterization is $\mathbf{W}_x = \mathbf{I}$, the matrix $\mathbf{W}_q$ is modified depending on the proximity to a singularity. As the measure of manipulability

$$w = \sqrt{\det(\mathbf{J}\mathbf{J}^T)}, \tag{3.5}$$

becomes zero at singularities, its absolute value serves at a distance measure from singularities [322]. The adaptation method for $\mathbf{W}_q$ then becomes

$$\mathbf{W}_q = \begin{cases} 0 & \text{for } w \geq w_0, \\ w_1(1 - \frac{w}{w_0})\mathbf{I} & \text{for } w < w_0, \end{cases} \tag{3.6}$$

with positive constants w_0, w_1. The proposed adaptation method leads to zero tracking error when being far away from singularities while damping the joint space velocity when being close to a singularity. If there are multiple tasks, they are concatenated in the vector $\mathbf{x}$ and prioritized by parameterizing the matrix $\mathbf{W}_x$ accordingly. Still this is only an approximate prioritization as possibly concurrent tasks are not executed in a hierarchical order. To account for a strict prioritization between two tasks $\dot{\mathbf{x}}_1, \dot{\mathbf{x}}_2$ with corresponding Jacobian matrices $\mathbf{J}_1, \mathbf{J}_2$, it is possible to project the joint velocity $\dot{\mathbf{q}}_2$ of the lower prioritized task in the null space of the Jacobian of the higher prioritized task as

$$\begin{aligned}\dot{\mathbf{q}} &= \mathbf{J}_1^+\dot{\mathbf{x}}_1 + (\mathbf{I} - \mathbf{J}_1^+\mathbf{J}_1)\dot{\mathbf{q}}_2, \\ \dot{\mathbf{q}}_2 &= \mathbf{J}_2^+\dot{\mathbf{x}}_2.\end{aligned}$$

Another more elaborate version is presented in [217], modifying the previous result such that the lower-prioritized task interacts as least as possible with the higher-prioritized one

$$\begin{aligned}\dot{\mathbf{q}} &= \mathbf{J}_1^+\dot{\mathbf{x}}_1 + (\mathbf{I} - \mathbf{J}_1^+\mathbf{J}_1)\dot{\mathbf{q}}_2, \\ \dot{\mathbf{q}}_2 &= \mathbf{J}_2^+(\dot{\mathbf{x}}_2 - \mathbf{J}_2\mathbf{J}_1^+\dot{\mathbf{x}}_1).\end{aligned}$$

From here on things become interesting:: Note that the pseudoinverse is only used for simplification during prioritization. It is also possible to use the DLS or WDLS solution. Now in the equation above the term $\mathbf{I} - \mathbf{J}_1^+ \mathbf{J}_1$ corresponds to the null space motion of the higher-prioritized task, i.e. the self-motion of the manipulator that does not interfere with the task. The approach can be cascaded to account for more than two prioritized tasks and calculation for this case is similar to (3.5).

Hence for motion imitation on a humanoid robot, the motion imitation task is in general lower prioritized than hardware-related constraints like balance of the robot, collision avoidance or joint limit avoidance. Let the secondary task thus be described by

$$\dot{\mathbf{q}}_2 = \mathbf{J}_{im} \dot{x}_{im},$$
$$\dot{x}_{im} = -K_{im} c_{im},$$

with the term c_{im} as the cost function encoding the imitation quality. Ergo the term c_{im} can be based on different measures, for example the squared distance in task space, the residual of a least squares solution or the error of a HMM. Through a gradient descent approach with variable gain factor K_{im} a task space velocity $\dot{\mathbf{x}}_{im}$ is calculated. The task space velocity is then mapped to joint pace by means of the imitation Jacobian $\mathbf{J}_{im}$, calculated as

$$\mathbf{J}_{im} = \left[\frac{\mathrm{d}\, c_{im}}{\mathrm{d}\, \dot{\mathbf{q}}} \right]^T.$$

En bloc the primary task achieves a COM-based balance, collision avoidance for a humanoid robot. Reactive fixed feet positions are also achived by prioritizing them all equally and concetanating all constraints::

$$\mathbf{J}_1 = \begin{bmatrix} \mathbf{J}_{com} & \mathbf{J}_{ca} & \mathbf{J}_f \end{bmatrix}^T,$$
$$\dot{\mathbf{x}}_1 = \begin{bmatrix} \dot{\mathbf{r}}_{com} & \dot{\mathbf{r}}_{ca} & \dot{r}_f \end{bmatrix}^T.$$

For sufficiently slow motions of the robot a zero moment point based calculation [258] is unnecessary. Instead, balance of the robot is achieved by keeping the projected COM $\mathbf{p}_{com} \in \mathbb{R}^2$ within the support polygon of the robot. A balance controller is used that moves the COM of the robot towards the center $\mathbf{p}_{com}^{ref} \in \mathbb{R}^2$ of the support polygon spanned by the two feet standing on the ground. Mathematically it is described by

$$\dot{\mathbf{x}}_{com} = K_{com}(\mathbf{p}_{com}^{ref} - \mathbf{p}_{com}),$$

with adjustable gain K_{com}.

Self-collision avoidance between two links of the robot is achieved through enclosing cylinders covering all robot links. In this case, if the distance $d_{i,ca}$ between two links falls below a certain threshold $d_{i,ca}^{min}$, the desired collision avoidance velocity $\dot{x}_{i,ca}$ is expressed as

$$\dot{x}_{i,ca} = K_{ca}(d_{i,ca}^{min} - d_{i,ca}) \text{ if } d_{i,ca} < d_{i,ca}^{min},$$

with gain factor K_{ca}, resulting in $\dot{\mathbf{x}}_{ca}$ and $\mathbf{J}_{ca}$ after concatenating multiple collision avoidance velocities. The same method also accounts for collision avoidance with obstacles in the environment. In the latter case the distance $d_{i,ca}$ is calculated between any robot link and the obstacle.

By defining three linearly independent sampling points $\mathbf{p}_{i,j,f}$ associated with the coordinate frame of each foot, a single Jacobian $\mathbf{J}_f$ accounting both for orientational and positional errors of the feet is calculated based on the cost function

$$c_f = \sum_{i=1}^{2}\sum_{j=1}^{3} \|\mathbf{p}_{i,j,f}^{ref} - \mathbf{p}_{i,j,f}\|_2^2.$$

The scalar $\dot{r}_f$ is defined in terms of the gradient descent speed $\dot{x}_f = -K_f c_f$ with gain K_f. Joint angle limits are considered through an underlying controller with bilateral boundaries. For a manipulator with a fixed stand, only collision avoidance and joint angle limits must be taken into account.

A highly redundant robot can fulfill the given imitation task perfectly while sufficing higher prioritized tasks. If the trajectory to be followed during the imitation task is calculated through LTE and the manipulator is not redundant enough to track it perfectly, LTE is used to adapt the optimal trajectory online. Fig. 3.1 shows an example where a planar 3DOF manipulator has to follow a straight trajectory. Whereas the left side shows the reference motion, the manipulator has to avoid an added obstacle on the right side with one of its joints, highlighted in red. Because the real EE trajectory deviates from the planned straight trajectory, LTE is used to replan new minimum acceleration trajectories online. Note that the replanned trajectories account only for the unconstrained case and are thus tracked perfectly after the manipulator does not collide anymore with the obstacle.

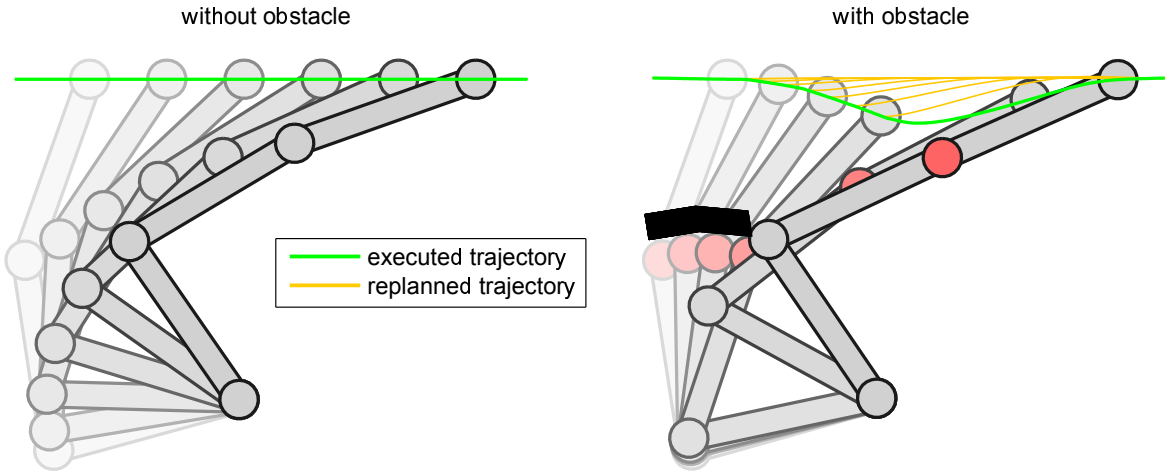


Figure 3.1: Prioritized inverse kinematics with continuous trajectory replanning. Left: Trajectory following without obstacle. Right: Trajectory following in the presence of an obstacle and with continuous replanning of optimal trajectories

3.4.2 LQR-based inverse kinematics with singularity avoidance

In this section the Jacobian-based IK scheme of the last section is combined with an LQR controller to consider not only the current manipulator's kinematic properties but

also future states and control inputs. A finite-horizon, discrete LQR controller assumes the robot kinematics to be constant over the prediction horizon and uses the resulting system dynamics as basis for a predictive controller. If the reference motion is both given in task- and joint space, singularity avoidance is achieved through an adaptive weighting scheme.

The discrete LQR controller for trajectory tracking

LQR control is a special class of predictive optimal control. Whereas in general such controllers are difficult to derive and can be often only approximated due to real-time constraints, a closed form solution exists for the LQR controller that can be derived analytically. There are four types of LQR controllers that have to be distinguished, depending on whether they consider a finite/infinite horizon or whether they work in the discrete/continuous domain. This section considers the discrete case with a finite horizon, yet the discrete case with an infinite horizon can be calculated in a similar manner. The LQR controller assumes that a discrete LTI system evolves according to

$$\mathbf{p}_{k+1} = \mathbf{A}\mathbf{p}_k + \mathbf{B}\mathbf{u}_k, \tag{3.7}$$

with the state vector $\mathbf{p}_k$, the input vector $\mathbf{u}_k$ and $\mathbf{A}, \mathbf{B}$ as the system respectively input matrix. This system representation is suitable for describing the convergence towards a fixed point. When following a trajectory, it is assumed that a reference state $\hat{\mathbf{p}}_k$ will be tracked perfectly using a reference input vector $\hat{\mathbf{u}}_k$ as

$$\hat{\mathbf{p}}_{k+1} = \mathbf{A}\hat{\mathbf{p}}_k + \mathbf{B}\hat{\mathbf{u}}_k. \tag{3.8}$$

Subtracting (3.7) from (3.8) yields the error dynamics which evolve similarly to the system dynamics for small errors as

$$(\mathbf{p}_{k+1} - \hat{\mathbf{p}}_{k+1}) = \mathbf{A}(\mathbf{p}_k - \hat{\mathbf{p}}_k) + \mathbf{B}(\mathbf{u}_k - \hat{\mathbf{u}}_k).$$

The cost function to be minimized by the LQR controller consists of both the state tracking error and the input tracking error over a number of future time steps, defined as

$$\begin{aligned} S_k &= \sum_{i=k}^{n} (\mathbf{p}_i - \hat{\mathbf{p}}_i)^T \mathbf{Q} (\mathbf{p}_i - \hat{\mathbf{p}}_i) + (\mathbf{u}_i - \hat{\mathbf{u}}_i)^T \mathbf{R} (\mathbf{u}_i - \hat{\mathbf{u}}_i), \\ &= \sum_{i=k}^{n} \Delta\mathbf{p}_i^T \mathbf{Q} \Delta\mathbf{p}_i + \Delta\mathbf{u}_i^T \mathbf{R} \Delta\mathbf{u}_i, \end{aligned} \tag{3.9}$$

with positive semidefinite weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ weighting the error in state and input space with respect to each other. Starting from a final state for $i = n$, the cost function in (3.9) is written as

$$S_n = \Delta\mathbf{p}_n^T \mathbf{Q}_f \Delta\mathbf{p}_n. \tag{3.10}$$

The key claim to be verified soon is that every cost function J_k can be written in the general form

$$S_k = \Delta\mathbf{p}_k^T \mathbf{V}_k \Delta\mathbf{p}_k,$$

in analogy to (3.10). Through backward induction the cost S_{k-1} is derived from S_k. Starting with the final cost function S_n, the cost function S_1 is obtained through the last backward induction step. Associated with the calculation of the cost function S_k is also the derivation of an optimal input vector $\mathbf{u}_k$ which gives the optimal input vector $\mathbf{u}_1$ for the current time step. The cost function S_{k-1} is split up into three parts as

$$\begin{aligned} S_{k-1} &= \Delta\mathbf{p}_{k-1}^T\mathbf{Q}\Delta\mathbf{p}_{k-1} + \Delta\mathbf{u}_{k-1}^T\mathbf{R}\Delta\mathbf{u}_{k-1} + S_k \\ &= \Delta\mathbf{p}_{k-1}^T\mathbf{Q}\Delta\mathbf{p}_{k-1} + \Delta\mathbf{u}_{k-1}^T\mathbf{R}\Delta\mathbf{u}_{k-1} + \\ &\quad (\mathbf{A}\Delta\mathbf{p}_{k-1} + \mathbf{B}\Delta\mathbf{u}_{k-1})^T\mathbf{V}_k(\mathbf{A}\Delta\mathbf{p}_{k-1} + \mathbf{B}\Delta\mathbf{u}_{k-1}), \end{aligned}$$

to account for the cost S_k and the cost to get from S_k to S_{k-1}. To find the optimal control input $\mathbf{u}_{k-1}$ for an optimal cost function S_{k-1}, the partial derivative is taken and set to zero

$$\begin{aligned} \frac{\partial S_{k-1}}{\partial \mathbf{u}_{k-1}} &= 2\mathbf{R}\Delta\mathbf{u}_{k-1} + 2\mathbf{B}^T\mathbf{V}_k(\mathbf{A}\Delta\mathbf{p}_{k-1} + \mathbf{B}\Delta\mathbf{u}_{k-1}) \\ &= 0. \end{aligned}$$

The optimal control input $\mathbf{u}_{k-1}$ is then calculated as

$$\begin{aligned} \mathbf{u}_{k-1} &= -(\mathbf{B}^T\mathbf{V}_k\mathbf{B} + \mathbf{R})^{-1}\mathbf{B}^T\mathbf{V}_k\mathbf{A}\Delta\mathbf{p}_{k-1} + \hat{\mathbf{u}}_{k-1} \\ &= \mathbf{K}_{k-1}\Delta\mathbf{p}_{k-1} + \hat{\mathbf{u}}_{k-1}, \end{aligned} \tag{3.11}$$

with $\mathbf{K}_{k-1}$ as the optimal LQR gain. Inserting $\mathbf{u}_{k-1}$ back into S_{k-1} yields

$$S_{k-1} = \Delta\mathbf{p}_{k-1}^T\mathbf{V}_{k-1}\Delta\mathbf{p}_{k-1},$$

with

$$\mathbf{V}_{k-1} = \left(\mathbf{Q} + \mathbf{K}_{k-1}^T\mathbf{R}\mathbf{K}_{k-1} + (\mathbf{A} + \mathbf{B}\mathbf{K}_{k-1})^T\mathbf{V}_k(\mathbf{A} + \mathbf{B}\mathbf{K}_{k-1})\right),$$

as the iterative update scheme to calculate $\mathbf{V}_{k-1}$ based on $\mathbf{V}_k$, verifying aforementioned claim. It is visible that neither the calculation of the optimal LQR gain nor of the optimal cost function depends on specific values for the system state or the system input. This leads to efficient calculation schemes as the both the optimal LQR gain and the optimal cost function have to be calculated only once for constant weighting matrices $\mathbf{Q}$ and $\mathbf{R}$.

Application to inverse kinematics

The LQR controller can be applied to IK in the field of motion imitation to track a given reference trajectory both in task- and joint space. An efficient singularity avoidance scheme is derived that lets the robot adapt a given motion to a different environment without getting trapped in a singular configuration. The differential robot kinematics in (3.1) are rewritten in discrete form as

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{J}_k\mathbf{dq}_k,$$

with $\mathbf{dq}_k$ as the joint angle difference between two subsequent sampling steps, $\mathbf{J}_k$ as the Jacobian matrix of the k-th time step and $\mathbf{p}_k$ as task space coordinates. The error dynamics are written in a similar way as

$$(\mathbf{p}_{k+1} - \hat{\mathbf{p}}_{k+1}) = (\mathbf{p}_k - \hat{\mathbf{p}}_k) + \mathbf{J}_k(\mathbf{dq}_k - \hat{\mathbf{dq}}_k),$$

and assume that a reference trajectory $\hat{\mathbf{p}}_k \in \hat{\mathbf{P}}$ with reference joint velocities $\hat{\mathbf{dq}}_k$ is given. The formulation in (3.12) is similar to (3.7) under one premise. The matrix $\mathbf{J}_k$ depends on the joint angles $\mathbf{q}$ and thus generally varies over time. In contrast the LTI system in (3.7) is per definition time independent. If the manipulator is far from a singular configuration and the joint angle velocities are small, the matrix $\mathbf{J}$ can be assumed to be constant over the prediction horizon as

$$\mathbf{J}_k = \mathbf{J} = const. \quad \forall k \in \{1, \dots, n\}.$$

Then the differential robot kinematics are written similar to (3.7) with

$$\mathbf{A} = \mathbf{I},$$
$$\mathbf{B} = \mathbf{J}.$$

The optimal input during backward induction is calculated in analogy to (3.11) as

$$\mathbf{dq}_{k-1} = -(\mathbf{J}^T\mathbf{V}_k\mathbf{J} + \mathbf{R})^{-1}\mathbf{J}^T\mathbf{V}_k(\mathbf{p}_k - \hat{\mathbf{p}}_k) + \hat{\mathbf{dq}}_{k-1}, \tag{3.12}$$

and shares common properties with the WDLS solution in (3.4) which is recapitulated as

$$\dot{\mathbf{q}} = (\mathbf{J}^T\mathbf{W}_x\mathbf{J} + \mathbf{W}_q)^{-1}\mathbf{J}^T\mathbf{W}_x\dot{\mathbf{x}}.$$

Whereas the cost minimized by the LQR controller for the error dynamics becomes

$$S_k = \sum_{i=k}^{n}(\mathbf{p}_i - \hat{\mathbf{p}}_i)^T\mathbf{Q}(\mathbf{p}_i - \hat{\mathbf{p}}_i) + (\mathbf{dq}_i - \hat{\mathbf{dq}}_i)^T\mathbf{R}(\mathbf{dq}_i - \hat{\mathbf{dq}}_i),$$

the WDLS solution minimizes a cost function of the form

$$S_{WDLS} = (\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}})^T\mathbf{W}_x(\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}) + \dot{\mathbf{q}}\mathbf{W}_q\dot{\mathbf{q}}.$$

Both cost functions are of similar structure, minimizing a weighted sum errors in task space and joint space. However, the LQR controller also considers future time steps whereas the WDLS solution only refers to a single time step. Both controllers also differ in the type of error to be minimized. The LQR cost function is based on a positional error in task space with respect to a reference trajectory and a velocity error in joint space with respect to a reference joint speed. The WDLS solution is based on a velocity error in task space between the desired velocity $\dot{\mathbf{x}}$ and the current velocity $\mathbf{J}\dot{\mathbf{q}}$. The term $\dot{\mathbf{q}}\mathbf{W}_q\dot{\mathbf{q}}$ is the velocity error in joint space with respect to zero velocity. Because the WDLS is solely based on differential errors, any imperfect solution with $S_{WDLS} \neq 0$ will lead to a persistent positional tracking error. As the LQR solution depends on the positional task space error, it overcomes this drawback. A related controller is presented in [33]. Based on the positional error both in joint space and task space, two possibly concurrent goals are considered. If the two reference trajectories in task space and joint space are conflicting, the used weighting scheme has to prioritize one over the other, leading to a persistent positional tracking error either in joint space or task space. Based on the four possible combinations of positional/velocity error in task space and positional/velocity error in

joint space, there exists a fourth type of controller based on the velocity error in task space and the positional error in joint space. Through a change of variables in (3.12) between joint space and task space coordinates, it can be derived in a similar manner. However, as most tasks are specified through positional task space errors, e.g. a minimum distance to an obstacle or zero distance to an grasped object, the benefit of using a velocity task space error in combination with a positional joint space error is questionable. To avoid singularities, an adaptive weighting scheme between task space and joint

method	LQR	WDLS	Calinon	hypoth. controller
positional task space error	✓		✓	
velocity task space error		✓		✓
positional joint space error			✓	✓
velocity joint space error	✓	✓		
predictive controller	✓			✓
persistent tracking error		✓	(✓)	

Table 3.1: Comparison between different controllers for motion imitation and adaptation depending on the underlying imitation measure

space objectives is proposed. Similar to WDLS the manipulability measure w is chosen such that close to a singularity the joint space objective is weighted more. In analogy to (3.6) this results in a modification of the weighting matrix $\mathbf{R}$ as

$$\mathbf{R} = \begin{cases} w_\epsilon \mathbf{I} & \text{for } w \geq w_0, \\ w_\epsilon \mathbf{I} + w_r (1 - \frac{w}{w_0}) \mathbf{I} & \text{for } w < w_0, \end{cases} \tag{3.13}$$

with w_r, w_ϵ as positive scalars. Note that w_ϵ is only required to guarantee theoretical stability of the resulting controller but can be set to zero for practical purposes. In addition, the task space objective is weighted less. The weighting is chosen such that in a singular configuration the weighting matrix $\mathbf{Q}$ becomes zero.

$$\mathbf{Q} = \begin{cases} w_q \mathbf{I} & \text{for } w \geq w_0, \\ w_q \frac{w}{w_0} \mathbf{I} & \text{for } w < w_0. \end{cases} \tag{3.14}$$

Two limiting cases are derived. Away from a singularity, i.e. $w \geq w_0$, the two weighting matrices are given as $\mathbf{R} = w_\epsilon \mathbf{I}$, $\mathbf{Q} = w_q \mathbf{I}$ and both the optimal input and the cost function update rule result in

$$\begin{aligned} \mathbf{dq}_{k-1} &= -(\mathbf{J}^T \mathbf{V}_k \mathbf{J} + w_\epsilon \mathbf{I})^{-1} \mathbf{J}^T \mathbf{V}_k (\mathbf{p}_k - \hat{\mathbf{p}}_k) + \hat{\mathbf{dq}}_{k-1}, \\ \mathbf{V}_{k-1} &= \left(w_q \mathbf{I} + w_\epsilon \mathbf{K}_{k-1}^T \mathbf{K}_{k-1} + (\mathbf{A} + \mathbf{B}\mathbf{K}_{k-1})^T \mathbf{V}_k (\mathbf{A} + \mathbf{B}\mathbf{K}_{k-1}) \right). \end{aligned}$$

In a singular configuration given by $w = 0$, the weighting matrices become $\mathbf{R} = (w_r + w_\epsilon)\mathbf{I}$, $\mathbf{Q} = 0$, which give

$$\begin{aligned} \mathbf{dq}_{k-1} &= -(\mathbf{J}^T \mathbf{V}_k \mathbf{J} + (w_r + w_\epsilon)\mathbf{I})^{-1} \mathbf{J}^T \mathbf{V}_k (\mathbf{p}_k - \hat{\mathbf{p}}_k) + \hat{\mathbf{dq}}_{k-1}, \\ \mathbf{V}_{k-1} &= \left((w_r + w_\epsilon) \mathbf{K}_{k-1}^T \mathbf{K}_{k-1} + (\mathbf{A} + \mathbf{B}\mathbf{K}_{k-1})^T \mathbf{V}_k (\mathbf{A} + \mathbf{B}\mathbf{K}_{k-1}) \right). \end{aligned}$$

For $w_\epsilon \approx 0$, it reduces to

$$\mathbf{dq}_{k-1} \approx \hat{\mathbf{dq}}_{k-1},$$
$$\mathbf{V}_{k-1} \approx \mathbf{0},$$

that is a direct feedthrough of the reference joint velocity. Close to a singularity both WDLS and the proposed controller weight the positional error in task space less in order to avoid the singularity. As the second objective of the WDLS controller is minimal joint velocity, the manipulator will move unnecessarily slow. In contrast the proposed controller will give more weight to the joint velocity of the reference motion, passing the singularity quickly.

Evaluation

The presented LQR approach is evaluated through simulations and compared with a WDLS scheme and an iterative linear quadratic regulator (ILQR) controller. With its assumption of a constant system and input matrix over the predicted horizon, it is located in between a non-predictive controller which optimizes only over a single time step and an ILQR approach which explicitly considers the time-varying nature of the underlying optimization problem [191]. Whereas the WDLS scheme is presented in Sec. 3.4.1, the ILQR approach is explained in this paragraph. It iteratively evaluates a trajectory $\mathbf{p}_k$ by applying the input $\mathbf{dq}_k$ based on (3.4.2), linearizes the system dynamics around the given trajectory and calculates an optimized local input through LQR backward induction. When applied to IK, the differential robot kinematics in (3.12) are rewritten as

$$\mathbf{p}_{k+1} = f(\mathbf{p}_k, \mathbf{dq}_k) = \mathbf{p}_k + \mathbf{J}(\mathbf{p}_k)\mathbf{dq}_k.$$

After approximating the system dynamics by a Taylor expansion around $\mathbf{p}_k^*, \mathbf{dq}_k^*$, we obtain

$$\mathbf{p}_{k+1} = f(\mathbf{p}_k^*, \mathbf{dq}_k^*) + \frac{\partial f(\mathbf{p}_k^*, \mathbf{dq}_k^*)}{\partial \mathbf{dq}_k}(\mathbf{dq}_k - \mathbf{dq}_k^*) + \frac{\partial f(\mathbf{p}_k^*, \mathbf{dq}_k^*)}{\partial \mathbf{p}_k}(\mathbf{p}_k - \mathbf{p}_k^*),$$

which results in the error dynamics as

$$\begin{aligned}(\mathbf{p}_{k+1} - \hat{\mathbf{p}}_{k+1}) = &\frac{\partial f(\mathbf{p}_k^*, \mathbf{dq}_k^*)}{\partial \mathbf{dq}_k}(\mathbf{dq}_k - \hat{\mathbf{dq}}_k) + \frac{\partial f(\mathbf{p}_k^*, \mathbf{dq}_k^*)}{\partial \mathbf{dq}_k}(\hat{\mathbf{dq}}_k - \mathbf{dq}_k^*) + \mathbf{f}(\mathbf{p}_k^*, \mathbf{dq}_k^*) - \hat{\mathbf{p}}_{k+1} \\ &+ \frac{\partial f(\mathbf{p}_k^*, \mathbf{dq}_k^*)}{\partial \mathbf{p}_k}(\mathbf{p}_k - \hat{\mathbf{p}}_k) + \frac{\partial f(\mathbf{p}_k^*, \mathbf{dq}_k^*)}{\partial \mathbf{p}_k}(\hat{\mathbf{p}}_k - \mathbf{p}_k^*),\end{aligned}$$

Through augmentation of the system state, the error dynamics can be rewritten in LQR form as

$$\begin{bmatrix} \mathbf{p}_{k+1} - \hat{\mathbf{p}}_{k+1} \\ 1 \end{bmatrix} = \mathbf{A}_k \begin{bmatrix} \mathbf{p}_k - \hat{\mathbf{p}}_k \\ 1 \end{bmatrix} + \mathbf{B}_k(\mathbf{p}_k - \hat{\mathbf{p}}_k),$$

with

$$\mathbf{A}_k = \begin{bmatrix} \frac{\partial f(\mathbf{p}_k^*, \mathbf{dq}_k^*)}{\partial \mathbf{p}_k} & f(\mathbf{p}_k^*, \mathbf{dq}_k^*) - \hat{\mathbf{p}}_{k+1} + \frac{\partial f(\mathbf{p}_k^*, \mathbf{dq}_k^*)}{\partial \mathbf{dq}_k}(\hat{\mathbf{dq}}_k - \mathbf{dq}_k^*) + \frac{\partial f(\mathbf{p}_k^*, \mathbf{dq}_k^*)}{\partial \mathbf{p}_k}(\hat{\mathbf{p}}_k - \mathbf{p}_k^*) \\ 0 & 1 \end{bmatrix},$$
$$\mathbf{B}_k = \begin{bmatrix} \frac{\partial f(\mathbf{p}_k^*, \mathbf{dq}_k^*)}{\partial \mathbf{dq}_k} \\ 0 \end{bmatrix}.$$

Away from a singular configuration, the effect of $\frac{\partial \mathbf{J}(\mathbf{p}_k)}{\partial \mathbf{p}_k}$ is negligible, allowing one to rewrite $\mathbf{A}_k$ and $\mathbf{B}_k$ in simpler form as

$$\mathbf{A}_k = \begin{bmatrix} 1 & \mathbf{J}(\mathbf{p}_k)\mathbf{dq}_k - \hat{\mathbf{p}}_{k+1} + \hat{\mathbf{p}}_k \\ 0 & 1 \end{bmatrix},$$
$$\mathbf{B}_k = \begin{bmatrix} \mathbf{J}(\mathbf{p}_k) \\ 0 \end{bmatrix},$$

which is both used when calculating the updated trajectory $\mathbf{p}_k$ and during the LQR backward induction step.

Fig. 3.2 shows a simulation that compares the three WDLS, LQR and ILQR controllers. For both LQR and ILQR controller the singularity avoidance scheme according to (3.13) and (3.14) is used. The parameterization given in Tab. 3.2 ensures comparability of the three controllers. A planar three-link manipulator is moved with constant joint velocity to generate the reference trajectory $\hat{\mathbf{p}}_k$ (black) and the reference joint velocity $\hat{\mathbf{dq}}_k$ as shown on the left side. The task space consists of the EE position in x/y direction and the EE orientation. The reference trajectory is then shifted in task space by a constant factor along the x/y direction, resulting in the modified trajectory (blue) that must be followed in the middle picture. As the robot enters a kinematic singularity when following the modified trajectory it is unable to track the reference trajectory perfectly, leading to a non-negligible offset in task space. Two plots on the right side illustrate the positional offset and EE velocity over time when tracking the modified trajectory. Looking at the offset plot, it is visible that both LQR and ILQR controller have a larger offset than the WDLS controller out of a singular configuration due to a imperfectly matching reference input $\hat{\mathbf{dq}}$. On the other hand, the reference input helps at leaving a singular configuration quickly as it does not slow down the manipulator. Looking at the velocity plot, the WDLS controller has by far the largest EE velocity when leaving the singular configuration. It must be mentioned that the ILQR controller has a significantly higher computational complexity ($\approx 5s$) than the LQR controller ($\approx 100ms$) or the WDLS controller ($\approx 50ms$) for the given scenario. The presented controller in (3.12) is a redefined version of the generic LQR controller in (3.11) for which stability is guaranteed for the infinite horizon case. The weighting scheme in (3.13)-(3.14) does not influence the stability properties as stability is guaranteed for all values of $\mathbf{R}$ and $\mathbf{Q}$.

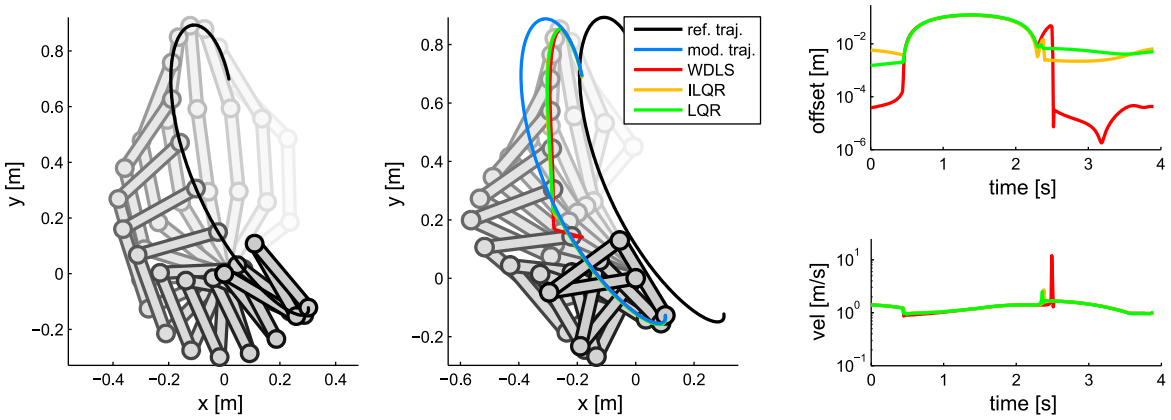


Figure 3.2: Evaluation of different singularity avoidance control schemes. Left: Original movement. Middle: Adapted movement. Right: Offset and velocity plots of the different control schemes for the adapted movement

parameter	value
w_0	0.05
w_1	1
w_q	1
w_r	1
w_ϵ	$< 1e-307$

Table 3.2: Parameters for the different singularity avoidance control schemes

3.4.3 Integrated motion planning and control by combining LQR and LTE

This section presents two approaches to combine LTE with an LQR controller to account for local trajectory properties not only during offline motion planning but also when controlling the robot online during execution. In combination with the IK approaches presented, this results in a whole-body motion control scheme letting a robot react instantly to sudden disturbances in an optimal manner.

State augmentation

One way to incorporate the local coordinates from LTE over the prediction horizon with an LQR controller is to augment both state and input vector. For the m-th order finite difference, not only the current state but also $m-1$ states of the past are considered in the state vector. This allows to include the local trajectory properties as

$$\begin{bmatrix} \mathbf{p}_{k+1} \\ \vdots \\ \mathbf{p}_{k-m-1} \\ \boldsymbol{\delta}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & \mathbf{A} & 0 \\ \mathbf{E}_0 & \dots & \mathbf{E}_m & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ \vdots \\ \mathbf{p}_{k-m} \\ \boldsymbol{\delta}_k \end{bmatrix} + \begin{bmatrix} \mathbf{B} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{B} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_k \\ \vdots \\ \mathbf{u}_{k-m} \end{bmatrix}, \tag{3.15}$$

with $\mathbf{p}_k, \mathbf{u}_k, \mathbf{A}, \mathbf{B}$ as the elements of the LTI system in (3.7) and the vector $\boldsymbol{\delta}_k$ as the local path properties for the LTI system. The matrices $\mathbf{E}_0, \ldots, \mathbf{E}_m$ account for the local trajectory properties, as such a common choice for second order finite differences is $\mathbf{E}_0 = \mathbf{I}, \mathbf{E}_1 = -2\mathbf{I}, \mathbf{E}_2 = \mathbf{I}$. To weight all three objectives, namely state error, control input error and error of the local trajectory properties, the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ in (3.9) are augmented accordingly, resulting in $\mathbf{Q}'$ and $\mathbf{R}'$

$$\mathbf{Q}' = \begin{bmatrix} \mathbf{Q} & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & \mathbf{Q} & 0 \\ 0 & 0 & 0 & \mathbf{F} \end{bmatrix}, \qquad \mathbf{R}' = \begin{bmatrix} \mathbf{R} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{R} \end{bmatrix}, \tag{3.16}$$

with the positive semidefinite matrix $\mathbf{F}$ accounting for the relative weight of the local path properties.

The principle of state augmentation is described in Fig. 3.3. Its schematic process flow is illustrated on the left side. For a given system, the desired reference state/trajectory is calculated a priori through LTE. The reference trajectory $\hat{\mathbf{p}} \in \hat{\mathbf{P}}$ serves as the input for the LQR controller to generate an optimal output trajectory $\mathbf{p} \in \mathbf{P}$. The right side shows the desired behavior in task space. For a nonzero weighting matrix $\mathbf{Q}'$ the resulting trajectory $\mathbf{P}$ converges to the reference trajectory $\hat{\mathbf{P}}$, illustrated by the mechanical analogy of a spring between the two trajectories. Due to the finite horizon method of LQR, augmenting the

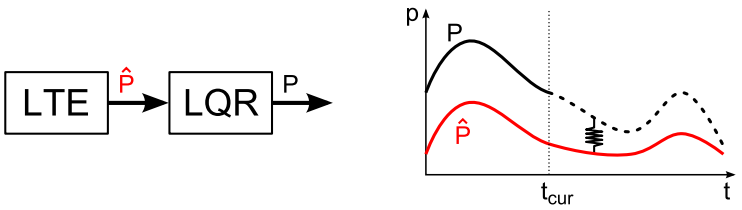


Figure 3.3: LQR/LTE state augmentation

state vector with local path properties will account for a number of path properties in the future given the history of the LTI system until the k-th time step. Though in general it results in slower convergence towards the reference path due to an additional constraint that must be considered, it makes it possible to consider local path properties also on a controller level. State augmentation cannot be used to consider LTE-specific constraints if they are not defined in a similar way for every sampling point of the trajectory (e.g. positional constraints for a few sampling points). This requires either a dynamic modification of the state vector, which is difficult and leads to discontinuities during execution or a conceptually different approach as described in the next paragraph. Stability can be guaranteed for the infinite-horizon case if the system in 3.15 is controllable [74].

Cost function augmentation

Augmenting the system state allows to combine the advantages of LTE with a predictive LQR controller in a straightforward fashion. When deviating from the reference trajec-

tory, the LQR controller calculates an optimal input given the system dynamics. Differing from the example in Fig. 3.1, the reference trajectory is calculated only once before execution and not updated through LTE at any time step. In contrast the method presented in this paragraph retargets the reference trajectory at every time step. In this case an alternative way of combining LTE and LQR based on an augmented LQR cost function is presented in this paragraph. Its scheme is depicted in Fig. 3.4. Similar to the normal LQR controller it is assumed that a fixed reference trajectory $\hat{\mathbf{P}}$ is given. A second LTE-deformed reference trajectory $\breve{\mathbf{P}}$ is calculated based on the system state for $t \leq t_{cur}$ and results in an optimal reference trajectory for all future time steps $t > t_{cur}$ for the current system state. The optimal input of the system then depends both on the state and input error with respect to $\hat{\mathbf{P}}$ and $\breve{\mathbf{P}}$. The resulting LQR cost function depending on the state and input error with respect to $\hat{\mathbf{p}}_i \in \hat{\mathbf{P}}$ and $\breve{\mathbf{p}}_i \in \breve{\mathbf{P}}$ is written as

$$\begin{aligned} S_k = \sum_{i=k}^{n} & (\mathbf{p}_i - \hat{\mathbf{p}}_i)^T \mathbf{Q} (\mathbf{p}_i - \hat{\mathbf{p}}_i) + (\mathbf{u}_i - \hat{\mathbf{u}}_i)^T \mathbf{R} (\mathbf{u}_i - \hat{\mathbf{u}}_i) + \\ & (\mathbf{p}_i - \breve{\mathbf{p}}_i)^T \mathbf{L} (\mathbf{p}_i - \breve{\mathbf{p}}_i) + (\mathbf{u}_i - \breve{\mathbf{u}}_i)^T \mathbf{M} (\mathbf{u}_i - \breve{\mathbf{u}}_i) \\ = \sum_{i=k}^{n} & \Delta\hat{\mathbf{p}}_i^T \mathbf{Q} \Delta\hat{\mathbf{p}}_i + \Delta\hat{\mathbf{u}}_i^T \mathbf{R} \Delta\hat{\mathbf{u}}_i + \Delta\breve{\mathbf{p}}_i^T \mathbf{L} \Delta\breve{\mathbf{p}}_i + \Delta\breve{\mathbf{u}}_i^T \mathbf{M} \Delta\breve{\mathbf{u}}_i \end{aligned} \tag{3.17}$$

with weighting matrices $\mathbf{Q}, \mathbf{R}, \mathbf{L}, \mathbf{M}$. Similar to the standard LQR problem, the cost function is written in a more general form as

$$\begin{aligned} S_k = & \mathbf{v}_{1,k}^T \Delta\hat{\mathbf{p}}_k + \mathbf{w}_{1,k}^T \Delta\breve{\mathbf{p}}_k + \\ & \Delta\hat{\mathbf{p}}_k^T \mathbf{v}_{2,k} + \Delta\breve{\mathbf{p}}_k^T \mathbf{w}_{2,k} + \\ & \Delta\hat{\mathbf{p}}_k^T \mathbf{X}_k \Delta\hat{\mathbf{p}}_k + \\ & \Delta\hat{\mathbf{p}}_k^T \mathbf{Y}_{1,k} \Delta\breve{\mathbf{p}}_k + \\ & \Delta\breve{\mathbf{p}}_k^T \mathbf{Y}_{2,k} \Delta\hat{\mathbf{p}}_k + \\ & \Delta\breve{\mathbf{p}}_k^T \mathbf{Z}_k \Delta\breve{\mathbf{p}}_k + c_k, \end{aligned}$$

with scalar c_k vectors $\mathbf{v}_{1,k}, \mathbf{w}_{1,k}, \mathbf{v}_{2,k}, \mathbf{w}_{2,k}$ and matrices $\mathbf{X}_k, \mathbf{Y}_{1,k}, \mathbf{Y}_{2,k}, \mathbf{Z}_k$. The backward induction step results in

$$
\begin{aligned}
S_{k-1} = {} & \Delta\hat{\mathbf{p}}_{k-1}^T \mathbf{Q} \Delta\hat{\mathbf{p}}_{k-1} + \Delta\hat{\mathbf{u}}_{k-1}^T \mathbf{R} \Delta\hat{\mathbf{u}}_{k-1} + \\
& \Delta\check{\mathbf{p}}_{k-1}^T \mathbf{L} \Delta\check{\mathbf{p}}_{k-1} + \Delta\check{\mathbf{u}}_{k-1}^T \mathbf{M} \Delta\check{\mathbf{u}}_{k-1} + \\
& J_k \\
= {} & \Delta\hat{\mathbf{p}}_{k-1}^T \mathbf{Q} \Delta\hat{\mathbf{p}}_{k-1} + \Delta\hat{\mathbf{u}}_{k-1}^T \mathbf{R} \Delta\hat{\mathbf{u}}_{k-1} + \\
& \Delta\check{\mathbf{p}}_{k-1}^T \mathbf{L} \Delta\check{\mathbf{p}}_{k-1} + \Delta\check{\mathbf{u}}_{k-1}^T \mathbf{M} \Delta\check{\mathbf{u}}_{k-1} + \\
& \mathbf{v}_{1,k}^T (\mathbf{A}\Delta\hat{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\hat{\mathbf{u}}_{k-1}) + \\
& \mathbf{w}_{1,k}^T (\mathbf{A}\Delta\check{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\check{\mathbf{u}}_{k-1}) + \\
& (\mathbf{A}\Delta\hat{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\hat{\mathbf{u}}_{k-1})^T \mathbf{v}_{2,k} + \\
& (\mathbf{A}\Delta\check{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\check{\mathbf{u}}_{k-1})^T \mathbf{w}_{2,k} + \\
& (\mathbf{A}\Delta\hat{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\hat{\mathbf{u}}_{k-1})^T \mathbf{X}_k (\mathbf{A}\Delta\hat{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\hat{\mathbf{u}}_{k-1}) + \\
& (\mathbf{A}\Delta\hat{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\hat{\mathbf{u}}_{k-1})^T \mathbf{Y}_{1,k} (\mathbf{A}\Delta\check{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\check{\mathbf{u}}_{k-1}) + \\
& (\mathbf{A}\Delta\check{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\check{\mathbf{u}}_{k-1})^T \mathbf{Y}_{2,k} (\mathbf{A}\Delta\hat{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\hat{\mathbf{u}}_{k-1}) + \\
& (\mathbf{A}\Delta\check{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\check{\mathbf{u}}_{k-1})^T \mathbf{Z}_k (\mathbf{A}\Delta\check{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\check{\mathbf{u}}_{k-1}) + \\
& c_{k-1}.
\end{aligned}
$$

The optimal input $\mathbf{u}_{k-1}$ is obtained by calculating the partial derivative of J_{k-1} and setting it to zero

$$
\begin{aligned}
\frac{\partial S_{k-1}}{\partial \mathbf{u}_{k-1}} = {} & 2\mathbf{R}\Delta\hat{\mathbf{u}}_{k-1} + 2\mathbf{M}\Delta\check{\mathbf{u}}_{k-1} + \\
& 2\mathbf{B}^T \mathbf{X}_k (\mathbf{A}\Delta\hat{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\hat{\mathbf{u}}_{k-1}) + \\
& 2\mathbf{B}^T \mathbf{Z}_k (\mathbf{A}\Delta\check{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\check{\mathbf{u}}_{k-1}) + \\
& \mathbf{B}^T (\mathbf{Y}_{1,k} + \mathbf{Y}_{2,k}) (\mathbf{A}\Delta\hat{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\hat{\mathbf{u}}_{k-1}) + \\
& \mathbf{B}^T (\mathbf{Y}_{1,k} + \mathbf{Y}_{2,k}) (\mathbf{A}\Delta\check{\mathbf{p}}_{k-1} + \mathbf{B}\Delta\check{\mathbf{u}}_{k-1}) + \\
& \mathbf{B}^T (\mathbf{v}_{1,k} + \mathbf{v}_{2,k} + \mathbf{w}_{1,k} + \mathbf{w}_{2,k}) \\
= {} & 0.
\end{aligned}
$$

This results in an optimal input as

$$\begin{aligned}\mathbf{u}_{k-1} = &\left(\mathbf{R}+\mathbf{M}+\mathbf{B}^T\mathbf{X}_k\mathbf{B}+\mathbf{B}^T\mathbf{Z}_k\mathbf{B}+\mathbf{B}^T(\mathbf{Y}_{1,k}+\mathbf{Y}_{2,k})\mathbf{B}\right)^{-1}\\ &\Bigg[\left(\mathbf{R}+\mathbf{B}^T\mathbf{X}_k\mathbf{B}+\frac{1}{2}\mathbf{B}^T(\mathbf{Y}_{1,k}+\mathbf{Y}_{2,k})\mathbf{B}\right)\hat{\mathbf{u}}_{k-1}+\\ &\left(\mathbf{M}+\mathbf{B}^T\mathbf{Z}_k\mathbf{B}+\frac{1}{2}\mathbf{B}^T(\mathbf{Y}_{1,k}+\mathbf{Y}_{2,k})\mathbf{B}\right)\check{\mathbf{u}}_{k-1}-\\ &\left(\mathbf{B}^T\mathbf{X}_k\mathbf{A}+\frac{1}{2}\mathbf{B}^T(\mathbf{Y}_{1,k}+\mathbf{Y}_{2,k})\mathbf{A}\right)\Delta\hat{\mathbf{p}}_{k-1}-\\ &\left(\mathbf{B}^T\mathbf{Z}_k\mathbf{A}+\frac{1}{2}\mathbf{B}^T(\mathbf{Y}_{1,k}+\mathbf{Y}_{2,k})\mathbf{A}\right)\Delta\check{\mathbf{p}}_{k-1}-\\ &\mathbf{B}^T(\mathbf{v}_{1,k}+\mathbf{v}_{2,k}+\mathbf{w}_{1,k}+\mathbf{w}_{2,k})\Bigg]\\ =&\mathbf{C}_1\hat{\mathbf{u}}_{k-1}+\mathbf{C}_2\check{\mathbf{u}}_{k-1}+\mathbf{C}_3\Delta\hat{\mathbf{p}}_{k-1}+\mathbf{C}_4\Delta\check{\mathbf{p}}_{k-1}+\mathbf{c}_5,\end{aligned} \tag{3.18}$$

from which the updated cost S_{k-1} is calculated as shown in App. B. After the optimal

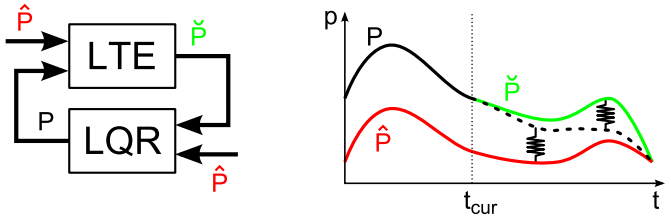


Figure 3.4: LQR/LTE cost function augmentation

input $\mathbf{u}_1$ is calculated and applied, the reference trajectory $\check{\mathbf{p}} \in \check{\mathbf{P}}$ is updated through LTE to provide an optimal reference trajectory for all future time steps $t > q$ given the evolution of the system for $t \leq q$. This is done by applying positional constraints (2.5) to all sampling points for $t \leq q$ to match the trajectory $\check{\mathbf{p}}$ with the past states of the system as shown in Fig. 3.4.

The LQR cost function results in a blending between the two trajectories. The relation between the weighting matrices $\mathbf{Q}$ and $\mathbf{L}$ and between $\mathbf{R}$ and $\mathbf{M}$ achieves a tradeoff of the convergence speed towards $\hat{\mathbf{P}}$ and the maintenance of the optimized local coordinates in $\check{\mathbf{P}}$. Two extrema can be analyzed: Setting $Ł = 0$, $\mathbf{M} = 0$ cancels the influence of the LTE-deformed trajectory, thus the LQR controller depends solely on $\hat{\mathbf{P}}$ and coincides with (3.9). Setting $\mathbf{Q} = 0$, $\mathbf{R} = 0$ ignores the reference trajectory $\hat{\mathbf{P}}$ which can be interpreted as a reference input to the feedback control system, thus convergence to the reference trajectory is either very slow or nonexistant.

The cost augmentation approach provides an integrated approach for combined motion planning (LTE) and motion control (LQR). Compared to the state augmentation approach this gives the user more freedom when designing a controller as the trajectory $\check{\mathbf{P}}$ can be retargeted through LTE in an optimal manner, thus considering not only the local coordinates but also other objectives as shown in Sec. 2. As such collision avoidance can be prioritized dynamically through intelligent reweighting of the matrices $Ł, \mathbf{M}, \mathbf{Q}, \mathbf{R}$

in highly cluttered environments whereas in free space more weight can be given on a quick convergence towards the reference trajectory.

Alternative calculation

Updating the trajectory $\check{\mathbf{P}}$ through LTE for the cost augmentation approach means to apply positional constraints to all sampling points for $t \leq q$. The two LTE updates between $t = t_{cur}$ to $t = t_{cur} + \Delta t$ differ by a single equation, the added positional constraint for $\check{\mathbf{p}}_{t_{cur}+\Delta t}$. In this case alternative approaches than the batch LS solution as in (2.4) can be more efficient. One option is to use recursive least squares (RLS) [115] which assumes that the solution for $\mathbf{z}_q$ of the linear system

$$\mathbf{y}_q = \mathbf{H}_q \mathbf{z}_q,$$

has to be updated due to a new measurement as

$$\begin{bmatrix} \mathbf{y}_q \\ y_{q+1} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_q \\ \mathbf{h}_{q+1}^T \end{bmatrix} \mathbf{z}_{q+1},$$

which is rewritten as

$$\mathbf{y}_{q+1} = \mathbf{H}_{q+1} \mathbf{z}_{q+1}.$$

Then an updated solution for $\mathbf{z}_{q+1}$ based on $\mathbf{z}_q$ is calculated through RLS as

$$\begin{aligned} \mathbf{k}_{q+1} &= \mathbf{E}_q \mathbf{h}_{q+1} (\mathbf{h}_{q+1}^T \mathbf{E}_q \mathbf{h}_{q+1} + 1)^{-1}, \\ \mathbf{z}_{q+1} &= \mathbf{z}_q + \mathbf{k}_{q+1} (\mathbf{y}_{q+1} - \mathbf{h}_{q+1}^T \mathbf{z}_q), \\ \mathbf{E}_{q+1} &= (\mathbf{I} - \mathbf{k}_{q+1} \mathbf{h}_{q+1}^T) \mathbf{E}_q, \end{aligned}$$

with the optimal gain $\mathbf{k}_{q+1}$, the correction factor $(\mathbf{y}_{q+1} - \mathbf{h}_{q+1}^T \mathbf{z}_q)$ and the error covariance matrix $\mathbf{E}_{q+1}$. Differing from the batch LS solution no matrix inversion is required. There are other incremental methods relying on subspace projection like Kaczmarz's algorithm [17] or exploiting a special structure of the underlying problem (lattice least squares [185, 94, 259], fast transversal algorithms [59]). All of them have a computational complexity of $\mathcal{O}(n)$ and are magnitudes faster than the pseudoinverse solution. Yet subspace projection methods are inapplicable as they minimize

$$\|\mathbf{z}_{q+1} - \mathbf{z}_q\|^2,$$

instead of

$$\|\mathbf{y}_{q+1} - \mathbf{H}_{q+1} \mathbf{z}_{q+1}\|.$$

Lattice least squares and fast transversal algorithms are theoretically applicable to LTE as they are specifically designed for time-evolving systems. The major difference between the two methods is that fast transversal based algorithms are of fixed order i.e. only applicable to trajectories with n sampling points, whereas lattice least squares algorithms are order-recursive i.e. the trajectory length can change online from n to $n+1$. Yet it is unknown whether and how both algorithms can be modified to include the local trajectory properties.

Evaluation

The cost function augmentation approach is compared with a standard LQR controller as shown in Fig. 3.5. The scenario consists of a particle with finite mass subject to friction and moving according to the dynamics in (3.19).

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 - \frac{D_p \Delta t}{M_p} \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\Delta t}{M_p} \end{bmatrix} u_k, \qquad \begin{bmatrix} x_0 \\ \dot{x}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{3.19}$$

The used parameterization is given in Tab. 3.3. Whereas the plots on the left side show the results for the standard LQR controller, the right side shows results for the cost function augmentation approach. Two plots on the top display the x-axis over time, showing both the desired trajectory (red), the resulting trajectory of the particle (black) and the planned optimal LTE trajectory at a few time steps (green). The lower two plots show the spatial offset between the resulting trajectory and the desired trajectory (red) and between the resulting trajectory and the planned optimal trajectory (green) at every time step. Both controllers behave roughly similar, yet the cost function augmentation approach allows to consider secondary objectives like collision avoidance when updating the planned trajectory.

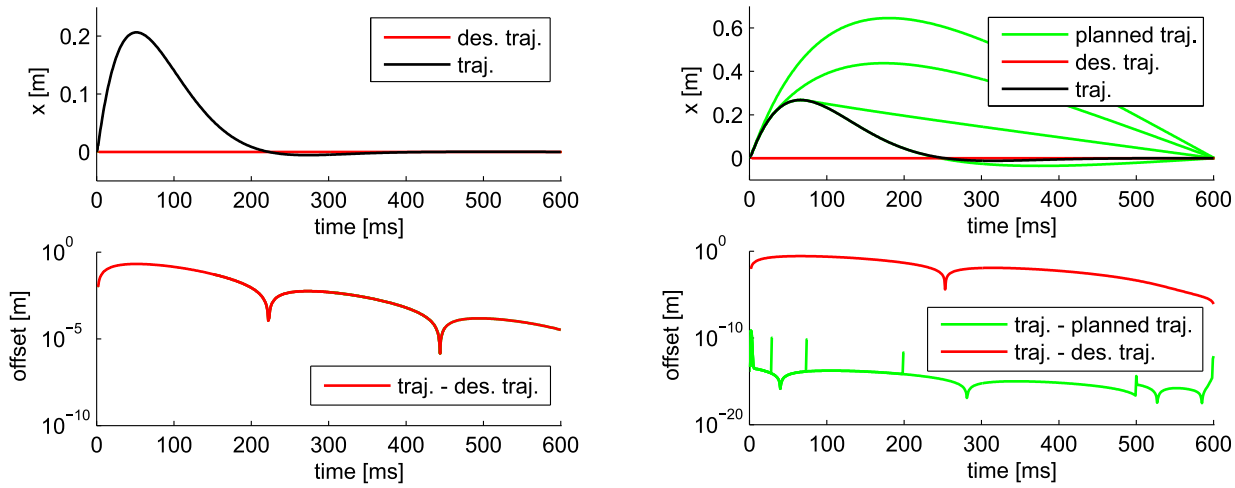


Figure 3.5: Evaluation of the cost function augmentation scheme and comparison with a standard LQR controller. Left: Standard LQR controller. Right: Cost function augmentation scheme. Top: Spatial extent. Bottom: Spatial offset

Stability

Theorem 2 Let the combined controller be defined according to (3.17) and the two trajectories $\check{\mathbf{P}}$ and $\hat{\mathbf{P}}$ coincide to a fixed point. Then asymptotic stability can be guaranteed for the infinite horizon case if the system $\{\mathbf{A}, \mathbf{B}\}$ is controllable.

parameter	value
Δt	0.01
M_p	20
D_p	5
$\mathbf{Q}, \mathbf{L}$	diag$(1e4, 1e3)$
$\mathbf{R}, \mathbf{M}$	diag$(1, 1)$

Table 3.3: Parameters for the cost function augmentation scheme

Proof. Stability for the infinite-horizon case of the standard discrete LQR controller can be guaranteed for controllable systems [74]. When augmenting the cost function, the interlocking scheme of LQR and LTE makes it difficult to guarantee stability. A special case is given for $\hat{\mathbf{p}} = \check{\mathbf{p}}$, i.e. if the fixed reference trajectory matches the LTE-deformed trajectory. Then the cost function in (3.17) reduces to

$$J_k = \sum_{i=k}^{n} (\mathbf{p}_i - \hat{\mathbf{p}}_i)^T (\mathbf{Q} + \mathbf{L})(\mathbf{p}_i - \hat{\mathbf{p}}_i) + (\mathbf{u}_i - \hat{\mathbf{u}}_i)^T (\mathbf{R} + \mathbf{M})(\mathbf{u}_i - \hat{\mathbf{u}}_i),$$

which is the standard discrete LQR controller. Convergence of $\check{\mathbf{p}}$ towards $\hat{\mathbf{p}}$ in the asymptotic case is ensured either through positional constraints for all future sampling points with increasing weighting factor $\boldsymbol{\Omega}$ over time or through a defined upper bound of the form

$$\|\check{\mathbf{p}}(t > t_{cur}) - \hat{\mathbf{p}}(t > t_{cur})\| \leq \chi e^{-t},$$

with $\check{\mathbf{p}}(t > t_{cur}), \hat{\mathbf{p}}(t > t_{cur})$ as all future sampling points of the reference and LTE-deformed trajectory and χ as a task-specific constant.

■

3.5 Upper deformation bounds satisfying dynamic constraints

This section derives upper deformation bounds that allow the robot to suffice given dynamic constraints. Because LTE deforms trajectories in task space, it cannot be used directly for dynamic constraints in joint space. A function is necessary that links the upper bounds on the applicable joint torques to the amount of deformation, resulting in a verification scheme whether the desired trajectory can be executed or not. The dynamic model of a given robot is described by

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}, \tag{3.20}$$

with $\mathbf{M}$ as the inertia matrix, $\mathbf{C}$ as the Coriolis and centrifugal forces, $\mathbf{g}$ accounting for all gravitational terms and $\boldsymbol{\tau}$ as the torque vector to achieve a desired joint movement $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$. The vectors $\mathbf{q}, \dot{\mathbf{q}}$ are related to the task space coordinates through the (differential) inverse kinematics scheme as in (3.1). After derivation we obtain

$$\ddot{\mathbf{x}} = \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\ddot{\mathbf{q}}, \tag{3.21}$$

respectively

$$\ddot{\mathbf{q}} = \mathbf{J}^{+}(\ddot{\mathbf{x}} - \dot{\mathbf{J}}\dot{\mathbf{q}}), \tag{3.22}$$

which link the task space acceleration $\ddot{\mathbf{x}}$ to the joint space velocity and acceleration $\dot{\mathbf{q}}, \ddot{\mathbf{q}}$. In general the joint torques are limited be the used hardware, thus the goal is to keep the actual joint torques below a specified threshold $\boldsymbol{\tau}_l$ as

$$|\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})| \leq \boldsymbol{\tau}_l. \tag{3.23}$$

On a more general scale the goal becomes to keep the norm of all torque values below a specific constant τ_{max}. This is done by evaluating the worst case scenario, i.e. by deriving upper bounds for each factor involved in the torque calculation.

Theorem 3 Let the dynamic model of the robot be given according to (3.20) and (3.21) and its kinematic model according to (3.1). Then the norm of the joint torques can be bounded from above by a scalar τ_{max} depending on the LTE deformation expressed by Bernstein polynomials according to (2.27).

Proof. The norm of the matrices $\mathbf{M}, \mathbf{C}, \mathbf{g}$ is limited from above by a constant factor [58] as

$$\begin{aligned} \|\mathbf{M}\| &\leq m_{max}, \\ \|\mathbf{C}\| &\leq c_{max}\|\dot{\mathbf{q}}\|, \\ \|\mathbf{g}\| &\leq g_{max}, \end{aligned}$$

thus (3.23) is rewritten as

$$m_{max}\|\ddot{\mathbf{q}}\| + c_{max}\|\dot{\mathbf{q}}\|\|\dot{\mathbf{q}}\| + g_{max} \leq \tau_l,$$

with the scalar value τ_{max} accounting for the limiting motor torque constraint. Using (3.1) and (3.22), the norms of $\ddot{\mathbf{q}}, \dot{\mathbf{q}}$ are be bounded by

$$\begin{aligned} \|\ddot{\mathbf{q}}\| &\leq \|\mathbf{J}^{+}\|(\|\ddot{\mathbf{x}}\| + \|\dot{\mathbf{J}}\|\|\dot{\mathbf{q}}\|), \\ \|\dot{\mathbf{q}}\| &\leq \|\mathbf{J}^{+}\|\|\dot{\mathbf{x}}\|. \end{aligned}$$

As $\dot{\mathbf{q}}, \ddot{\mathbf{q}}$ depend both on the Jacobian $\mathbf{J}$ and the tasks space velocity/acceleration $\dot{\mathbf{x}}, \ddot{\mathbf{x}}$, their upper bounds must be determined too. Due to the finite extent of each robot, the norm of the manipulator Jacobian is bounded from above by a constant scalar. If the Jacobian is Lipschitz continuous and the joint velocities are bounded, the norm of $\dot{\mathbf{J}}$ is bounded by a scalar factor j_{max} as

$$\|\dot{\mathbf{J}}\| \leq j_{max}.$$

An upper bound of the norm $\mathbf{J}^{+}$ can be derived for certain cases through the singular values $\sigma_{min}, \ldots, \sigma_{max}$ of $\mathbf{J}$. It is

$$\|\mathbf{J}^{+}\|_2 \leq \frac{1}{\sigma_{min}} = s_{max},$$

with finite scalar s_{max} if the robot does not enter a singular configuration, which can be ensured through a workspace analysis.

In most cases upper bounds of $\dot{\mathbf{x}}, \ddot{\mathbf{x}}$ cannot be calculated in closed form and must be determined by evaluating the deformed trajectory. An exception is a spline-based deformation, see Sec. 2.6.2. When splitting the deformed trajectory up in two parts {1} and {2}, the sole transition of {1} enables one to calculate an upper bound of the norm of $\dot{\mathbf{x}}, \ddot{\mathbf{x}}$ for {1} independent of the deformation, resulting in $\dot{x}_{max}, \ddot{x}_{max}$. If the additive deformation term {2} is represented by Bernstein polynomials $\dot{\mathbf{B}}_k^l$, its derivative can be calculated based on

$$\dot{\mathbf{B}}_k^l = l(\mathbf{B}_{k-1}^{l-1} - \mathbf{B}_k^{l-1}).$$

For a spline curve described by

$$\mathbf{x} = \sum_{k=0}^{l} \mathbf{a}_k \mathbf{B}_k^l,$$

its velocity is calculated as

$$\dot{\mathbf{x}} = l \sum_{k=0}^{l-1} (\mathbf{a}_{k+1} - \mathbf{a}_k) \mathbf{B}_k^{l-1},$$

and the acceleration as

$$\ddot{\mathbf{x}} = l(l-1) \sum_{k=0}^{l-2} (\mathbf{a}_{k+2} - 2\mathbf{a}_{k+1} + \mathbf{a}_k) \mathbf{B}_k^{l-2},$$

Thanks to the non-negativity of each Bernstein polynomial, the norm of $\dot{\mathbf{x}}, \ddot{\mathbf{x}}$ can be bounded by

$$\begin{aligned} \|\dot{\mathbf{x}}\| &\leq \dot{x}_{max} + l \max_{k=0,\ldots,l-1} \|\mathbf{a}_{k+1} - \mathbf{a}_k\|, \\ \|\ddot{\mathbf{x}}\| &\leq \ddot{x}_{max} + l(l-1) \max_{k=0,\ldots,l-2} \|\mathbf{a}_{k+2} - 2\mathbf{a}_{k+1} + \mathbf{a}_k\|. \end{aligned} \tag{3.24}$$

This results in the torque bound τ_{max} as

$$m_{max} s_{max} (\|\ddot{\mathbf{x}}\| + j_{max} s_{max} \|\dot{\mathbf{x}}\|) + c_{max} s_{max}^2 \|\dot{\mathbf{x}}\| \|\dot{\mathbf{x}}\| + g_{max} \leq \tau_{max},$$

with $\|\ddot{\mathbf{x}}\|, \|\dot{\mathbf{x}}\|$ bounded by according to (3.24).

■

3.6 Whole-body control through task-space distance meshes

This section presents a novel encoding scheme for motion imitation by considering not only the robot pose but also its interactions with objects in the environment. Differing from LTE which is mainly used to encode and reproduce EE trajectories, the appraoch provides a holistic framework to handle both free space motions and close contact scenarios. Based on a probabilistic encoding using HMMs, the importance of each task imitation aspect is automatically extracted and considered during motion adaptation and control.

Information encoding and reproduction

Similar to the previous chapter the robot consists of a number of k joints concatenated in the joint space vector $\mathbf{q}$ and a set of l links with uniquely identifiable position in task space $\mathbf{x}_i^l \in \mathbb{R}^3, i = \{1, \ldots, l\}$. The same accounts for a set of objects in the environment. In order to obtain information about the objects' orientation, o object points $\mathbf{x}_i^o \in \mathbb{R}^3, i = \{1, \ldots, o\}$ are defined depending on the objects' geometry. We define $n = l + o$ as the total number and $\mathbf{x}_i \in \mathbb{R}^3, i = \{1, \ldots, n\}$ as the total set of position vectors. The time-varying version of $\mathbf{x}_i$ at time step t is denoted $\mathbf{x}_{i,t}$, the one of $\mathbf{q}$ is denoted $\mathbf{q}_t$. Then the distance mesh vector $\mathbf{d} \in \mathbb{R}^{n(n-1)/2}$ comprises every possible Euclidean distance between the position vectors. For every time step $t \in \{0, \ldots, t_{max}\}$ a new distance mesh vector $\mathbf{d}_t$ is calculated, thus encoding the temporal progress of the distance mesh. To obtain the task-relevant elements of the distance mesh, multiple demonstrations of a task are performed and the elements of the resultant distance meshes for each demonstration are processed using a continuous hidden Markov model (CHMM) and GMR. The CHMM is represented by the parameter set $\{\pi, \mathbf{a}, \mathbf{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ with π for the initial state probabilities, $\mathbf{a}$ the state transition probabilities and $\mathbf{w}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ for weight, mean and covariance of each state of the CHMM. In addition to the spatial mean $\mu_{s,g}$ and variance $\Sigma_{s,g}$ for every Gaussian mixture component g for a single dimension, the temporal information are used to calculate the temporal mean $\mu_{t,g}$, the temporal variance $\Sigma_{t,g}$ and the covariance between temporal and spatial data $\Sigma_{ts,g}$. Differing from existing approaches in literature [292] the interest of this approach is solely on motion imitation and adaptation, thus no movement recognition is performed. Through concatenation of all elements the motion is encoded in the spatio-temporal mean vector $\boldsymbol{\mu}_g$ and covariance matrix $\boldsymbol{\Sigma}_g$. For a single dimension of the CHMM for a single state it is defined as

$$\boldsymbol{\mu}_g = \begin{bmatrix} \mu_{s,g} \\ \mu_{t,g} \end{bmatrix}, \qquad \boldsymbol{\Sigma}_g = \begin{bmatrix} \Sigma_{s,g} & \Sigma_{ts,g} \\ \Sigma_{ts,g} & \Sigma_{t,g} \end{bmatrix}.$$

If all task repetitions are matching in the temporal domain one can also use GMM in combination with an EM algorithm to extract the task-relevant information. The retrieval of a most likely motion for straightforward motion imitation is done through GMR based on $\boldsymbol{\mu}_g$ and $\boldsymbol{\Sigma}_g$ for every dimension as described in [34], resulting in a reference distance mesh $\hat{\mathbf{d}}_t \in \mathbb{R}^{n(n-1)/2}$ together with its associated reference spatial variance $\hat{\boldsymbol{\Sigma}}_{s,t} \in \mathbb{R}^{n(n-1)/2 \times n(n-1)/2}$ as

$$\hat{\boldsymbol{\Sigma}}_{s,t} = \mathrm{diag}(\Sigma_{s,g,1}, \ldots, \Sigma_{s,g,n(n-1)/2}),$$

The combination of $\hat{\mathbf{d}}_t$, $\hat{\boldsymbol{\Sigma}}_{s,t}$ and $\mathbf{x}_{i,t}$ provides all information about the robot's pose and its interaction with objects in the environment. Even if $\mathbf{x}_{i,t}$ is not given fully but one knows only the position of objects in the environment, the most likely position of all robot links is reconstructed through numeric optimization. The time-dependent variance $\hat{\boldsymbol{\Sigma}}_{s,t}$ encodes the relative importance of each element of the reference distance mesh. In this sense a large variance corresponds to a low importance of the specific element and a low variance corresponds to a high importance, except for those elements of the distance mesh where

the robot's geometry imposes a time-invariant constant distance. Whereas generally the inverse of the variance as

$$\mathbf{W}_t = \hat{\mathbf{\Sigma}}_{s,t}^{-1},$$

is used to represent the variance-dependent importance matrix $\mathbf{W}_t \in \mathbb{R}^{n(n-1)/2}$ of every element of the distance mesh, a more general version is used in this paper as

$$\mathbf{W}_t = f(\hat{\mathbf{\Sigma}}_{s,t}^{-1}),$$

such that $\mathbf{W}_t$ is positive definite. As the motion representation solely depends on elements in task space, the challenge of weighting joint space and task space elements relatively to each other is overcome. The initial inclusion of all possible task space distances also omits the need to specify task-relevant elements manually.

When mapping the motion on the robot and executing the encoded motion in a changed environment, the reproduced distance mesh $\tilde{\mathbf{d}}_t(\mathbf{q}_t)$ at time t generally differs from $\hat{\mathbf{d}}_t$. To find an optimal robot configuration defined by the joint angles $\mathbf{q}_t$, a cost function c_{cf} based on the weighted difference between $\tilde{\mathbf{d}}_t(\mathbf{q}_t)$ and the reference distance mesh $\hat{\mathbf{d}}_t$ is minimized as

$$\begin{aligned}
\mathbf{d}_{rel[i]} &= \frac{\tilde{\mathbf{d}}_{t[i]}(\mathbf{q}_t) - \hat{\mathbf{d}}_{t[i]}}{\hat{\mathbf{d}}_{t[i]}}, \\
c_{cf} &= \mathbf{d}_{rel}^T \mathbf{W}_t \mathbf{d}_{rel}, \\
\mathbf{q}_{min} &= \underset{\mathbf{q}_t}{\operatorname{argmin}}(c_{cf}).
\end{aligned}$$

Differing from [48] the relative deviation $(\tilde{\mathbf{d}}_{t[i]}(\boldsymbol{\theta}_t) - \hat{\mathbf{d}}_{t[i]})/\hat{\mathbf{d}}_{t[i]}$ is used instead of the absolute deviation $\tilde{\mathbf{d}}_{t[i]}(\boldsymbol{\theta}_t) - \hat{\mathbf{d}}_{t[i]}$ to focus on deviations of small distance mesh elements, i.e. close contact situations. For execution on a real robot the cost function c_{cf} is optimized using a gradient descent approach as described in Sec. 3.4.1.

Handling modeling/measurement errors

Using a task-space measure is challenging as it does not consider the underlying kinematic properties of the robot. This can lead to kinematic singularities or impossible robot configurations as indicated in Fig. 3.7. Shown are two different manipulators, one element of the distance mesh and a hypothetical manipulability ellipsoid measuring the amount of manipulability. If caused by a modeling/measurement error the distance between base and EE has to be increased when descending the gradient of c_{cf}, joint angle velocities are minimal for the left manipulator. In contrast the right configuration will enter a singularity as the manipulator reaches its kinematic workspace limits. If the DLS or WDLS method is used, their additive term during matrix inversion will cause an unpredictable drift of the joint angles close to a singularity. Thus another option based on the manipulability ellipsoid is proposed, modifying the cost function c_{cf} such that elements of the distance mesh leading to singular configurations are weighted less. The manipulability ellipsoid is defined as

$$\dot{\mathbf{x}}^T (\mathbf{J}\mathbf{J}^T)^{-1} \dot{\mathbf{x}} = 1,$$

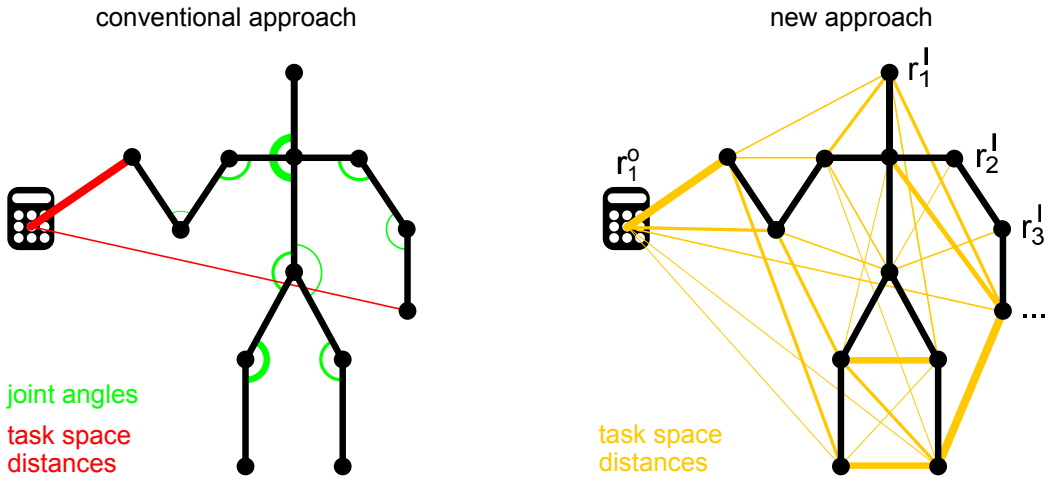


Figure 3.6: Information encoding and reproduction for task-space distance meshes. Left: Conventional approach encoding joint angles and only selected task space distances. Right: Proposed approach encoding the full set of task space distances

and used to calculate the manipulability along an arbitrary dimension in task space. Yet changing the elements of the distance mesh only influences the manipulability along a single direction. If $\dot{\mathbf{r}}_{i,t} \in \mathbb{R}^3$ is the direction the i-th element of $\mathbf{d}_t$ accounts for, we get

$$\dot{\mathbf{x}}_{i,t}^T (\mathbf{J}_{i,t} \mathbf{J}_{i,t}^T)^{-1} \dot{\mathbf{x}}_{i,t} = 1,$$

for every element of $\mathbf{d}_t$. It is then

$$\mathbf{D}_{man} = \text{diag}(\left\|\dot{\mathbf{x}}_{1,t}\right\|_2, \ldots, \left\|\dot{\mathbf{x}}_{n(n-1)/2,t}\right\|_2),$$
$$\mathbf{W}_t = \mathbf{D}_{man} \hat{\mathbf{\Sigma}}_{s,t}^{-1},$$

resulting in a weighting scheme that reduces the influence of distance mesh elements leading to a singular configuration during gradient descent.

To calculate the elements of $\mathbf{D}_{man}$, a Jacobian matrix must be calculated between any two links of the robot. In general Jacobians are calculated with respect to a base link 0, resulting in ${}^0\mathbf{J}_G$ and ${}^0\mathbf{J}_F$ for two different links G and F. The calculation of ${}^F\mathbf{J}_G$ given only ${}^0\mathbf{J}_G$ and ${}^0\mathbf{J}_F$ is [289]

$${}^F\mathbf{J}_{G,tr} = \mathbf{R}_0 \left\{ {}^0\mathbf{J}_{G,tr} - {}^0\mathbf{J}_{F,tr} + ({}^0\mathbf{x}_G - {}^0\mathbf{x}_F)^\times \, {}^0\mathbf{J}_{F,\omega} \right\},$$

with ${}^0\mathbf{r}_G$ and ${}^0\mathbf{r}_F$ as the position of link G and F with respect to the base link 0, the subscript ${}_{tr}$ respectively ${}_\omega$ denoting the translational and rotational part of the Jacobian, the rotation matrix $\mathbf{R}_0$ accounting for the pose of the base link with respect to world coordinates and ${}^\times$ for the cross product and therefore skew-symmetric matrix of a vector.

Complexity reduction

Depending on the number of position vectors the calculation of the distance mesh and gradient computation has an $\mathcal{O}(n^2)$ complexity. In addition, the weighting matrix is highly

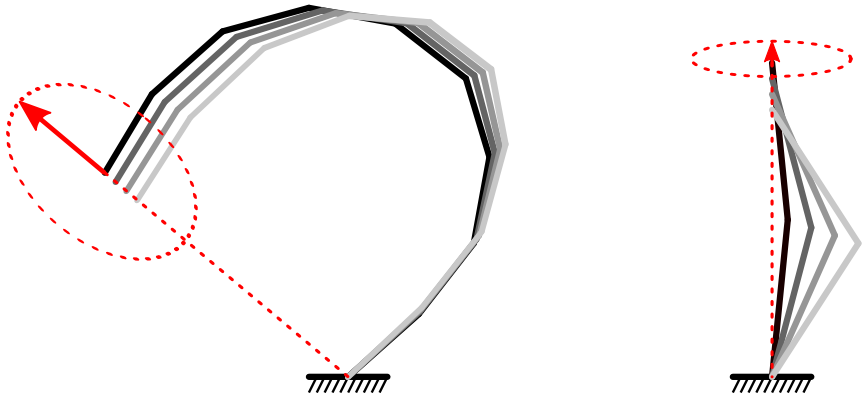


Figure 3.7: Handling modeling/measurement errors. Changing the distance between EE and base causes much higher joint velocities for the right configuration than for the left one. To consider the effect, the manipulability in direction of each mesh distance element is calculated (red arrow)

redundant as its elements vary by several magnitudes. Two methods are presented that overcome both problems by considering only a subset of all distance mesh elements when calculating the cost function c_{cf}. The first, simple method considers only a specific portion of the largest elements in $\mathbf{W}_t$. Yet this can cause undesired null space movements of the robot if c_{cf} is not calculated based on the postions of all links. Thus an elaborate method interprets the distance mesh as a mechanical system of connected springs keeping the points $\mathbf{x}_{i,t}$ in position. Through a successive removal of springs it is investigated how well the points $\mathbf{x}_{i,t}$ are fixed to a defined spatial position. This is done by looking at the volume of the unit energy ellipsoid of every point. Differing from the first method it considers not only the weighting $\mathbf{W}_t$ but also the spatial arrangement of the distance mesh elements. The j-th element of c_{cf} is now seen as a mechanical spring with

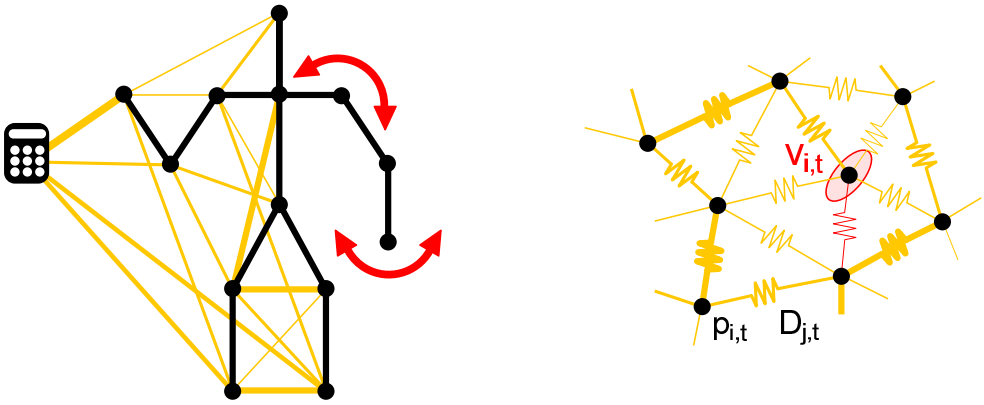


Figure 3.8: Complexity Reduction. Left: Undesired null space movements of the robot in case the cost function c_{cf} does not depend on the position of all links. Right: Interpretation of the distance mesh as a mechanical system of connected springs

displacement

$$\mathbf{s}_{j,t} = \tilde{\mathbf{d}}_{j,t}(\boldsymbol{\theta}_t) - \hat{\mathbf{d}}_{j,t},$$

and spring constant

$$D_{j,t} = \frac{\mathbf{W}_{jj,t}}{\hat{\mathbf{d}}_{j,t}^T \hat{\mathbf{d}}_{j,t}}.$$

For a perfect motion reproduction it is $\mathbf{s}_{j,t} = 0 \ \forall j,t$. In case of only a single spring between two endpoints and a infinitesimal small displacement of one of the two endpoints $\mathbf{x}_{i,t}$ in a random direction $\Delta\mathbf{s}_{i,t}$, the displacement force $\mathbf{f}_{i,t} \in \mathbb{R}^3$ is expressed as

$$\mathbf{f}_{i,t} \approx \mathbf{D}_{j,t} \Delta\mathbf{s}_{i,t}.$$

Note that the symmetric matrix $\mathbf{D}_{j,t} \in \mathbb{R}^{3\times3}$ is used instead of the scalar $D_{j,t}$, accounting for the spatial alignment of $\Delta\mathbf{s}_{i,t}$ and $\hat{\mathbf{d}}_{j,t}$. Consequently, the energy $E_{i,t}$ required to cause the displacement is written as

$$E_{i,t} = \frac{1}{2} \Delta\mathbf{s}_{i,t}^T \mathbf{D}_{j,t} \Delta\mathbf{s}_{i,t}.$$

If there is more than one spring attached to the point $\mathbf{x}_{i,t}$, the displacement energy becomes the sum

$$E_{i,t} = \frac{1}{2} \Delta\mathbf{s}_{i,t}^T (\sum_j \mathbf{D}_{j,t}) \Delta\mathbf{s}_{i,t}. \tag{3.25}$$

In (3.25) it is visible that the energy required to displace the point $\mathbf{x}_i$ depends both on the direction of displacement and the amount of displacement. To find redundant configurations where a displacement of $\mathbf{x}_i$ is possible without change of the energy $E_{i,t}$ we look at the size of the constant energy ellipsoid defined by

$$1 = \frac{1}{2} \Delta\mathbf{s}_{i,t}^T (\sum_j \mathbf{D}_{j,t}) \Delta\mathbf{s}_{i,t},$$

for every point $\mathbf{x}_{i,t}$. Similar to the manipulability ellipsoid, its volume $v_{i,t}$ is calculated up to a constant multiplying factor as

$$v_{i,t} = \sqrt{\det\left((\sum_j \mathbf{D}_{j,t})^{-1} \right)}.$$

The final cost function $c_{r,t}$ is then defined as the sum over all volumes $v_{i,t}$ by

$$c_{r,t} = \sum_{i=1}^{n} v_{i,t}.$$

During the selection process only the elements minimizing c_r are chosen.

The matrix $\mathbf{D}_{j,t}$ is the combination of the scalar stiffness value $D_{j,t}$ and a matrix $\mathbf{M}$ only considering the amount of $\Delta\mathbf{s}_{i,t}$ in direction of $\hat{\mathbf{d}}_{j,t}$. Using the rotation matrix $\mathbf{R}_{j,t}$ that coaligns the vector $[1,0,0]^T$ with $\hat{\mathbf{d}}_{j,t}$, it is calculated as

$$\mathbf{D}_{j,t} = D_{j,t}\mathbf{R}_{j,t}\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\mathbf{R}_{j,t}^T.$$

Whereas in theory the subset of distance mesh elements has to be recalculated at every time step, it leads to discontinuities during gradient computation as the distance mesh constantly changes its appearance. A more robust, yet approximate method is to calculate only a single subset based on the summed values of $\mathbf{W}_t$ and $\hat{\mathbf{d}}_t$ over time, resulting in the cost function c_r. Note that Delaunay tetrahedralization as used originally in [120] is suboptimal as it does not consider the weighting $\mathbf{W}_t$ for all possible distance mesh elements.

3.7 Endeffector control using piecewise linear systems

Presented in this section is a motion control scheme which - when combined with LTE - ensures quick motion retargeting while ensuring convergence of the manipulator's integral curves towards the reference trajectory. In addition it allows a smooth variation of different execution speeds, overcoming problems arising from using a discrete reference trajectory. By creating a set of PL systems depending on the underlying trajectory's shape and interpolating smoothly between them, a continuous vector field is created around a discrete reference trajectory. The method shares similarities with [235], yet it does not depend on a least squares solution to calculate the vector field. Differing from other approaches like [33, 260], the set of PL systems does not need to be learned from multiple demonstrations but is calculated based on a single reference trajectory. If the trajectory is deformed, it is updated quickly. Convergence of the integral curves along the trajectory towards an end point is ensured by a smooth interpolation between multiple first order dynamical systems.

PL system representation

Similar to LTE the reference trajectory is represented by a set of sampling points and temporal information as as $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n]^T$. The trajectory consists of trajectory segments τ_i defined as the line segment between two subsequent sampling points $\mathbf{p}_i, \mathbf{p}_{i+1}$, $i \in \{1, \ldots, n-1\}$. Every trajectory segment has a mid point $\mathbf{p}_{m,i} = \frac{\mathbf{p}_i + \mathbf{p}_{i+1}}{2}$, lying on the hyperplane h_i with corresponding normal vector $\mathbf{n}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}$. The PL systems for the remainder of this paper are of the general form

$$\dot{\mathbf{x}} = \mathbf{A}_i\mathbf{x} + \mathbf{b}_i,$$

with $\mathbf{A} \in \mathbb{R}^{m\times m}, \mathbf{b} \in \mathbb{R}^m$. For the i-th trajectory segment, they are split up as

$$\begin{aligned}\mathbf{A}_i &= \mathbf{A}_{\|i} + \mathbf{A}_{\perp i},\\ \mathbf{b}_i &= \mathbf{b}_{\|i} + \mathbf{b}_{\perp i},\end{aligned}$$

consisting of a parallel PL system $V_{\|i} = \{\mathbf{A}_{\|i}, \mathbf{b}_{\|i}\}$ responsible for moving an object along the trajectory and an orthogonal PL system $V_{\perp i} = \{\mathbf{A}_{\perp i}, \mathbf{b}_{\perp i}\}$ for convergence towards the trajectory in case of a sudden disturbance, see Fig. 3.9. In order to calculate the elements of the PL systems, an orthonormal basis in $\mathbb{R}^m$ with basis vectors $\mathbf{a}_{j,i},\ j \in \{1,\ldots,m\}$ is defined such that $\mathbf{a}_{1,i}$ coaligns with $\mathbf{p}_{i+1} - \mathbf{p}_i$. It is assumed that the eigenvectors $\mathbf{e}_{j,i}$ of $\mathbf{A}_i$ coincide with $\mathbf{a}_{j,i}$. The scalar values $\lambda_{\|j,i}$ and $\lambda_{\perp j,i}$ denote the eigenvalues of $\mathbf{A}_{\|i}$ and $\mathbf{A}_{\perp i}$. Then the parallel PL system $V_{\|i}$ has to fulfill the conditions

$$\begin{aligned}\lambda_{\|1,i} = \lambda_{\|2,i} = \ldots = \lambda_{\|m,i} &= 0,\\ \dot{\mathbf{x}}(\mathbf{p}_{m,i}) = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{t_{i+1} - t_i} &> 0,\end{aligned} \tag{3.26}$$

that is a PL system with constant velocity parallel to the i-th trajectory segment. The orthogonal PL system $V_{\perp i}$ has to fulfill the condition

$$\begin{aligned}\lambda_{\perp 1,i} &= 0,\\ \lambda_{\perp 2,i} = \lambda_{\perp 3,i} = \ldots = \lambda_{\perp m,i} &< 0,\\ \dot{\mathbf{x}}(\mathbf{p}_{m,i}) &= 0,\end{aligned} \tag{3.27}$$

i.e. the line through $\mathbf{p}_i, \mathbf{p}_{i+1}$ being the only stable set of equilibrium points. By defining $\mathbf{A}_{\|i}$ and $\mathbf{b}_{\|i}$ as

$$\begin{aligned}\mathbf{A}_{\|i} &= \mathbf{0},\\ \mathbf{b}_{\|i} &= \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{t_{i+1} - t_i},\end{aligned}$$

it suffices (3.26). A heuristic approach is used to find a solution that suffices (3.27). By remembering that

$$\dot{\mathbf{x}} = \mathbf{A}_{\perp i}\mathbf{x} + \mathbf{b}_{\perp i},$$

one can create the equation system

$$\begin{aligned}\dot{\mathbf{x}}(\mathbf{p}_{m,i}) &= 0,\\ \dot{\mathbf{x}}(\mathbf{p}_{m,i} + \mathbf{a}_{1,i}) &= 0,\\ \dot{\mathbf{x}}(\mathbf{p}_{m,i} + \mathbf{a}_{q,i}) &= -\kappa_i\, \mathbf{a}_{q,i}, \quad \forall q \in \{2,\ldots,m\}\end{aligned}$$

and solve it for the variables in $\mathbf{A}_{\perp i}$ and $\mathbf{b}_{\perp i}$ in order to construct the orthogonal PL system. The strength parameter $\kappa_i > 0$ adjusts the influence of the orthogonal PL system $V_{\perp,i}$, accounting for the rate at which any object converges back to the trajectory after a disturbance. Note that the norm of any vector $\mathbf{v}_{\perp i}$ caused by the PL system $V_{\perp i}$ is directly proportional to the minimal distance $d_{m,i}$ to the line through $\mathbf{p}_{i+1}, \mathbf{p}_i$ as

$$\|\mathbf{v}_{\perp i}\| = d_{m,i}\kappa_i. \tag{3.28}$$

An interpolation scheme according to (3.7) with the weighting factor $w_i \in [0,1]$ blending over the PL systems of subsequent trajectory segments is proposed in order to avoid discontinuities

$$\dot{\mathbf{x}} = w_i(\mathbf{A}_i + \mathbf{b}_i) + (1-w_i)(\mathbf{A}_{i+1} + \mathbf{b}_{i+1}), \quad = w_i(\mathbf{v}_{\|i} + \mathbf{v}_{\perp i}) + (1-w_i)(\mathbf{v}_{\|i+1} + \mathbf{v}_{\perp i+1}).$$

with the vector $\mathbf{v}_{\|i}$ caused by the PL system $V_{\|i}$ The trajectory segment τ_i is defined by the two points $\mathbf{p}_i, \mathbf{p}_{i+1}$, the trajectory segment τ_{i+1} by $\mathbf{p}_{i+1}, \mathbf{p}_{i+2}$. Let the projection of an object position onto the τ_i segment be $\mathbf{p}_{p,i}$ and its relative length $r_i = \frac{\|\mathbf{p}_{p,i} - \mathbf{p}_i\|}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}$. The interpolation scheme weight w_i is given as

$$\begin{aligned} r_i' &= |r_i - 0.5|, \\ r_{i+1}' &= |r_{i+1} - 0.5|, \\ w_i &= \frac{r_{i+1}'}{r_i' + r_{i+1}'}, \end{aligned}$$

with the offset of 0.5 moving the moment of switching to the hyperplane h_i. Note that this interpolation scheme is only well defined for intersection angles γ_i

$$\gamma_i = \arccos\left(\frac{(\mathbf{p}_{i+2} - \mathbf{p}_{i+1}) \cdot (\mathbf{p}_{i+1} - \mathbf{p}_i)}{\|\mathbf{p}_{i+2} - \mathbf{p}_{i+1}\| \|\mathbf{p}_{i+1} - \mathbf{p}_i\|}\right) < \frac{\pi}{2}, \tag{3.29}$$

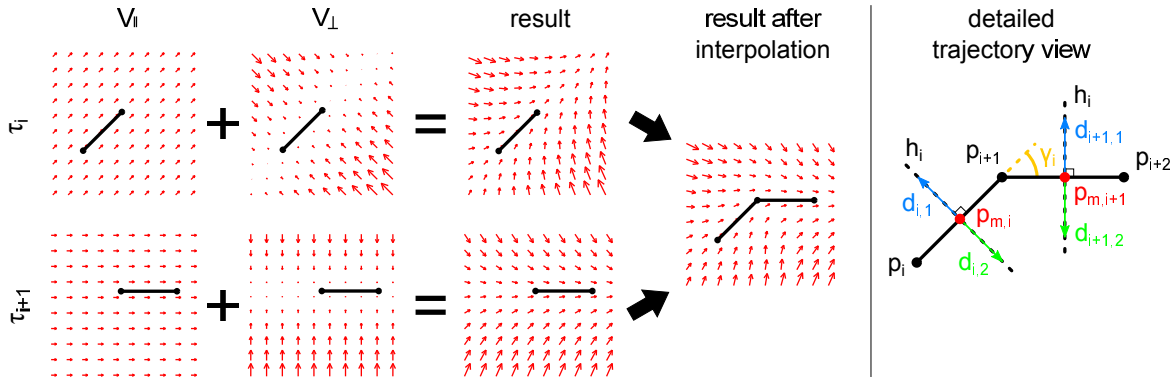


Figure 3.9: PL system overview. Left: Calculated vector field both after superposition and split up into individual parallel and orthogonal vector fields. Right: Schematic view

Convergence and stability

This paragraph deals with the question under which conditions convergence of the integral curves towards the reference trajectory can be ensured, thus corresponding to the stability properties of the PL system approach.

Theorem 4 Let the PL system be described according to (3.26)-(3.29). Then convergence of the integral curves towards the last sampling point $\mathbf{p}_n$ is guaranteed in an area around the trajectory where

$$\|\mathbf{v}_{\|i}\| \cos(\gamma_i) > d_{m,i}\kappa_i \sin(\gamma_i), \tag{3.30}$$

holds.

Proof. Two aspects of convergence must be differentiated, transition convergence and point convergence. Transition convergence is defined as the convergence from any point $\mathbf{s}$ either on h_i or in the area in between the two hyperplanes h_i, h_{i+1} defined by $\mathbf{n}_i \cdot (\mathbf{s} - \mathbf{p}_{m,i}) \geq 0$ and $\mathbf{n}_{i+1} \cdot (\mathbf{s} - \mathbf{p}_{m,i+1}) < 0$ to a point on the hyperplane h_{i+1}. Point convergence means that a point $\mathbf{s}$ either on the last hyperplane h_{n-1} or in one of the two half spaces defined by $\mathbf{n}_{n-1} \cdot (\mathbf{s} - \mathbf{p}_{m,n-1}) \geq 0$ eventually converges to the last sampling point $\mathbf{p}_n$. By combining both convergence properties it is ensured that a point moving along the trajectory eventually converges to $\mathbf{p}_n$. To prove transition convergence, the vector field $\dot{\mathbf{p}}(\mathbf{s})$ always has to point in direction of $\mathbf{n}_{i+1}$ as

$$\dot{\mathbf{x}}(\mathbf{s})^T \mathbf{n}_{i+1} > 0. \tag{3.31}$$

By decomposing $\dot{\mathbf{p}}(\mathbf{s})$ as

$$\dot{\mathbf{x}}(\mathbf{s}) = w_1(\mathbf{v}_{\|i} + \mathbf{v}_{\perp i}) + (1 - w_1)(\mathbf{v}_{\|i+1} + \mathbf{v}_{\perp i+1}),$$

it accounts both for the orthogonal/parallel components of the PL systems V_i, V_{i+1} and the interpolation scheme with weight w_1. Looking at the geometric properties, (3.31) is rewritten as

$$\begin{aligned} w_1(\|\mathbf{v}_{\|i}\| \cos(\gamma_i) - \|\mathbf{v}_{\perp i}\| \sin(\gamma_i)) + w_2\|\mathbf{v}_{\|i+1}\| > 0, \\ w_1(\|\mathbf{v}_{\|i}\| \cos(\gamma_i) - d_{m,i}\kappa_i \sin(\gamma_i)) + w_2\|\mathbf{v}_{\|i+1}\| > 0. \end{aligned} \tag{3.32}$$

The minimal values of (3.32) are given for $w_1 = 1$, resulting in

$$\|\mathbf{v}_{\|i}\| \cos(\gamma_i) > d_{m,i}\kappa_i \sin(\gamma_i),$$

which is a necessary condition for transition convergence. Depending on the intersection angle γ_i it defines an admissible region around the trajectory segment whose extent can approach ∞ for $\gamma_i \to 0$ in the limiting case or 0 for $\gamma_i \to \frac{\pi}{2}$. Point convergence for $\mathbf{p}_n$ is achieved by creating a PL system V for the last trajectory segment such that it fulfills

$$\begin{aligned} \lambda_{1,n} = \lambda_{2,n} = \ldots = \lambda_{m,n} < 0, \\ \dot{\mathbf{x}}(\mathbf{p}_n) = 0, \end{aligned}$$

that is a point attractor for $\mathbf{p}_n$ which is created following the idea in (3.7). The proof is analogous to the one for transition convergence with the only difference that $\|\mathbf{v}_{\|i+1}\|$ for the last trajectory segment is not constant anymore due to the point attractor. Yet as the (3.33) does not depend on the term $\|\mathbf{v}_{\|i+1}\|$, it also result in

$$\|\mathbf{v}_{\|i}\| \cos(\gamma_i) > d_{m,i}\kappa_i \sin(\gamma_i).$$

■

Continuity/differentiability

Due to the used interpolation scheme continuity and differentiability are ensured as long as subsequent hyperplanes h_i, h_{i+1} do not intersect with each other. Differentiability across the hyperplanes in the moment of switching is shown by calculating the derivative of the PL system at the left and right side of a point $\mathbf{p}_h$ on the hyperplane h_i. For the left side $\mathbf{p}_h^-$ it is with $w_{i-1} = 0, w_i = 1$

$$\left.\frac{\mathrm{d}\, w_{i-1}(\mathbf{A}_{i-1}\mathbf{p} + \mathbf{b}_{i-1}) + w_i(\mathbf{A}_i\mathbf{p} + \mathbf{b}_i)}{\mathrm{d}\,\mathbf{p}}\right|_{\mathbf{x}=\mathbf{p}_h^-} = \mathbf{A}_i.$$

With the weights $w_i = 1, w_{i+1} = 0$ for the right side $\mathbf{p}_h^+$ it is

$$\left.\frac{\mathrm{d}\, w_i(\mathbf{A}_i\mathbf{p} + \mathbf{b}_i) + w_{i+1}(\mathbf{A}_{i+1}\mathbf{p} + \mathbf{b}_{i+1})}{\mathrm{d}\,\mathbf{p}}\right|_{\mathbf{x}=\mathbf{p}_h^+} = \mathbf{A}_i,$$

thus ensuring continuity and differentiability across the hyperplanes and making the method applicable for a variety of tracking problems which require $\mathscr{C}^2$ continuity.

Bounding volume

It is possible to determine upper bounds for the maximal deviation from the reference trajectory as the object moves along the trajectory. These bounds depend solely on the spatial distance between two subsequent sampling points and the intersection angle γ_i and are thus simple to calculate. The upper bounds are calculated in terms of distance d_i to the mid point $\mathbf{p}_{m,i}$ when passing the hyperplane h_i, measuring how the distance d_i propagates to the distance d_{i+1} when passing the next hyperplane h_{i+1}. When considering only the two dimensions in which two subsequent trajectory segments are not collinear, one can define an outer side $d_{i,1}$ and an inner side $d_{i,2}$ for the distance d_i, see Fig. 3.9. In case of an offset $d_{i,1}$, the maximal propagated offset $d_{i+1,1}$ is defined with $l_1 = \frac{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}{2}$, $l_2 = \frac{\|\mathbf{p}_{i+2} - \mathbf{p}_{i+1}\|}{2}$ and the intersection angle γ_i as

$$d_{i+1,1} = \max\left(\frac{d_1}{\cos\gamma_i} + l_2 \tan\gamma_i, l_1 \sin\gamma_i\right). \tag{3.33}$$

In a similar fashion, the maximal propagated offset $d_{i+1,2}$ is defined as

$$d_{i+1,2} = \max(l_2 \tan\gamma_i, d_2 \cos\gamma_i + l_1 \sin\gamma_i). \tag{3.34}$$

In the general case one has to consider the worst case d_{i+1} for $d_i = \{d_1, d_2\}$ as

$$\begin{aligned} d_{i+1} &= \max(d_1', d_2'), \\ \frac{d_{i+1}}{d_i} &\geq 1. \end{aligned} \tag{3.35}$$

For values of γ_i close to $\frac{\pi}{2}$, the first terms of the *max* operator in (3.33) and (3.34) become dominant. By remembering that $\frac{1}{\cos\gamma_i} \approx \tan\gamma_i$ for $\gamma_i \approx \frac{\pi}{2}$, one can rewrite (3.35) as

$$\frac{d_{i+1}}{d_i} \sim \tan\gamma_i.$$

For values of γ_i close to 0, the second terms of the *max* operator in (3.33) and (3.34) become dominant. In this case, it is

$$\frac{d_{i+1}}{d_i} \approx 1.$$

So far, only the spatial bounding volume is considered. Another aspect is temporal boundedness, that is the time horizon within which the hyperplane h_{i+1} will be passed when starting from a point on the hyperplane h_i. It is for a given offset $d_i = \max\{d_1, d_2\}$ for the lower distance boundary $d_{l,i}$ and upper distance boundary $d_{u,i}$ orthogonal to h_{i+1} to be travelled

$$\begin{aligned} d_{u,i} &= l_1\cos\gamma_i + l_2 + d_1\sin\gamma_i, \\ d_{l,i} &= l_1\cos\gamma_i + l_2 - d_2\sin\gamma_i. \end{aligned}$$

In analogy to (3.32) velocity bounds for the minimal velocity $v_{l,i}$ and maximal velocity $v_{u,i}$ orthogonal to h_{i+1} are determined as

$$\begin{aligned} v_{u,i} &= \max(\|\mathbf{v}_{\|i}\|\cos(\gamma_i) + d_{m,i}\kappa_i\sin(\gamma_i), \|\mathbf{v}_{\|i+1}\|), \\ v_{l,i} &= \min(\|\mathbf{v}_{\|i}\|\cos(\gamma_i) - d_{m,i}\kappa_i\sin(\gamma_i), \|\mathbf{v}_{\|i+1}\|). \end{aligned}$$

Then the minimal time $t_{l,i}$ and maximal time $t_{u,i}$ required to travel from hyperplane h_i to h_{i+1} is calculated as

$$\begin{aligned} t_{u,i} &= \frac{d_{u,i}}{v_{l,i}}, \\ t_{l,i} &= \frac{d_{l,i}}{v_{u,i}}. \end{aligned}$$

As a result, one can calculate both temporal and spatial bounds for an object moving along the trajectory. The principle of spatial bounds along the hyperplanes can be extended to arbitrary hyperplanes, allowing one to cover the entire space.

Spatial bounds approximation

Whereas conservative spatial bounds of the deviation of the integral curve C from the reference path are presented which hold for any $\kappa_i > 0$, a good approximation can be made under the assumption of $\gamma_i \approx 0$ and $\|\mathbf{v}_{\|i+1}\| \approx \|\mathbf{v}_{\|i}\|$. Then for an object starting

from the hyperplane h_1 with an initial distance $d_{m,1}$ to the trajectory - see (3.28) - the resulting distance $d_{m,s}$ when passing the hyperplane h_s is calculated as

$$\begin{aligned} d_{m,s} &= d_{m,1} - \int_C V_\perp dx, \\ &\approx d_{m,1} \prod_{i=1}^{s} exp(-\kappa_i \Delta t_i), \\ &\approx d_{m,1} \prod_{i=1}^{s} exp(-\kappa_i \frac{\|\mathbf{p}_{m,i+1} - \mathbf{p}_{m,i}\|}{\|\mathbf{v}_{\|i}\|}), \end{aligned} \tag{3.36}$$

with $\int_C$ as the integral over the integral curve C and Δ_t as the time required to traverse from the hyperplane h_i to h_{i+1}.

3.8 Path similarity through subspace projection

This section introduces a generic classification framework to measure the similarity between two discretized paths. Based upon similar principles as LTE, it accounts for a wider class of paths as it is based upon a learning-free, hierarchical descriptor. Although different path similarity measures of LTE presented in Sec. 2.5 can also be used to compare two paths, they are only applicable under two restrictive conditions: Both path must consist of a similar number of sampling points and they must be free of noise. The first condition is obvious as the similarity measures in (2.18)-(2.19) either iterate over n sampling points or apply the Frobenius norm to the difference of two matrices with similar size. The second condition becomes clear by looking at (2.31). Depending on the derivative order of the local path properties, the sensitivity towards high-frequency noise increases accordingly. A solution is to define a similarity measure both on a local and global scale. Measurement noise will affect only local path properties whereas path deformations act on a global scale while maintaining the local path properties. A hierarchical approach iteratively splits up the path into smaller subpaths and captures the characteristic properties at each granularity, see [86].

Splitting process

The splitting process for a discretized path is driven by two main ideas. The first one is to split every path segment in exactly two different subpaths. When writing the dependence between paths and subpaths in a tree structure, it results in a full binary tree which captures the global path properties with the root node and the local path properties along the entire path with the leaf nodes. This can be ensured by splitting a path with n sampling points at the sampling point with index $\frac{n}{2}$ at depth $d = 1$, then at $\frac{n}{4}$ and $\frac{3n}{4}$ at depth $d = 2$, etc. Whereas the structural path properties are captured this way, the spatial distribution of sampling points along the trajectory has a huge influence on the spatial aspect of the splitting process. Thus the second idea is to maintain specific geometric properties of the path during every splitting process at the cost of an even split. Every path $\mathbf{P}$ consists of

a set of path segments $\mathbf{P}_{ij}$ containing all the sampling points $[\mathbf{p}_i, \mathbf{p}_{i+1}, \ldots, \mathbf{p}_j]^T$. At the first level of the splitting process, the path $\mathbf{P}_{1n} = \mathbf{P}$ has to be split up by a sampling point $\mathbf{p}_i \in \mathbf{P}_{1n}$ such that the two resulting lines $\mathbf{L}_{1i}$ through $(\mathbf{p}_1, \mathbf{p}_i)$ and $\mathbf{L}_{in}$ through $(\mathbf{p}_i, \mathbf{p}_n)$ achieve a tradeoff between resembling the path well and splitting it up evenly for $i = \frac{n}{2}$. Inspired by minimal surface energy considerations $\mathbf{p}_k^{proj}$ denotes the orthogonal projection of a sampling point $\mathbf{p}_k \in \mathbf{P}_{1i}$ on $\mathbf{L}_{1i}$. Then the weighted squared length L_{1i} of the trajectory and the sum of squares S_{1i} is calculated as

$$L_{1i} = m_L \left(\sum_{k=1}^{i-1} \left\| \mathbf{p}_{k+1} - \mathbf{p}_k, \right\| \right)^2,$$
$$S_{1i} = \sum_{k=1}^{i} \left\| \mathbf{p}_k - \mathbf{p}_k^{proj} \right\|^2, \tag{3.37}$$

with m_L as the number of sampling points of each path segment. The same calculation is done for any point $\mathbf{p}_k \in \mathbf{P}_{in}$ and the second line, resulting in S_{in} and L_{in}. The optimal sampling point $\mathbf{p}_s$, named splitting point, splitting the trajectory into two subtrajectories is

$$\mathbf{p}_s = \underset{p_i \in \mathbf{P}_{1n}}{\operatorname{argmin}}(L_{1i} + S_{1i} + L_{in} + S_{in}), \quad i = 1, \ldots, p. \tag{3.38}$$

The values S_{1i}, S_{in} and L_{1i}, L_{in} act as energy terms stored in two types of "mechanical springs". Minimizing S_{1i} and S_{in} means to minimize the squared distance between the two line segments and the path, thus resembling the path as good as possible in a least squared sense. Minimizing L_{1i} and L_{in} on the other hand centers the splitting point with respect to the number of sampling point n in case of equidistantly spaced sampling points. In all other cases it centers the splitting point with respect to the arc length of each path segment. The structural properties of every path segment $\mathbf{P}_{ij}$ are uniquely described by

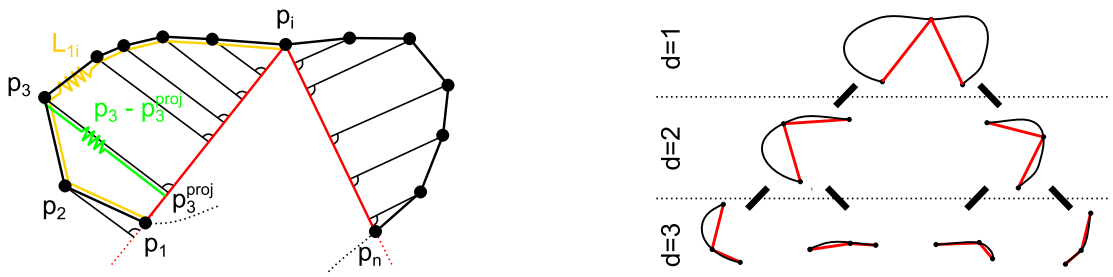


Figure 3.10: Shape tree creation overview. Left: Splitting process. Right: Resulting shape tree

its start point $\mathbf{p}_i$ and end point $\mathbf{p}_j$. When continuing iteratively with the splitting process, every path segment will consist only of two subsequent sampling points $(\mathbf{p}_{j-1}, \mathbf{p}_j)$ in the limiting case. All resulting path segments are stored in a binary tree called "shape tree" [86] together with a corresponding depth value $d \in \{1, \ldots, d_{max}\}$, see Fig. 3.10.

Similarity measure

When comparing two paths, their similarity is encoded in a single scalar value, the similarity measure. Similar to LTE its calculation is inspired by the Frenet-Serret formulas in (2.13). The Frenet-Serret formulas have an alternative interpretation as the offset of each point with respect to a lower-dimensional subspace. As such the velocity of a particle is interpreted as the distance to a 0D subspace (the current position) after a certain time and curvature is the offset related to a 1D subspace (a straight line). Yet the Frenet-Serret formulas cannot be applied in the generic case if multiple sampling points are not subsequent. Thus the interpretation of the subspace projection is generalized as

$$\begin{aligned} S_q &= \text{span}(\mathbf{p}_{s2} - \mathbf{p}_{s1}, \ldots, \mathbf{p}_{sq} - \mathbf{p}_{s1}), \quad q = 2, \ldots, m, \\ \mathbf{p}_s - \mathbf{p}_{s1} &= \mathbf{p}_{\parallel q} + \mathbf{p}_{\perp q}, \quad \mathbf{p}_{\parallel q} \in S_q, \mathbf{p}_{\perp q} \perp S_q, \\ l_{\perp q} &= \|\mathbf{p}_{\perp q}\|, \end{aligned} \tag{3.39}$$

with $\mathbf{p}_s$ being the splitting point, $\mathbf{p}_{s1}$ the start point and $\mathbf{t}_{s2}$ the end point of the trajectory segment. In case of a q-dimensional space first a set of bases S_q spanning different subspaces with dimension $1, \ldots, n-1$ are defined. Then the vector $\mathbf{p}_s - \mathbf{p}_{s1}$ is split up into a normal component $\mathbf{p}_{\parallel q} \in S_q$ and an orthogonal component $\mathbf{p}_{\perp q} \perp S_q$, see Fig. 3.11. The value $l_{\perp q}$ as the norm of the offset to each subspace forms the principal component to compare trajectories. For $m > 2$, additional sampling points beside $\mathbf{p}_{s2}$ and $\mathbf{p}_{s1}$ as stated in (3.39) must be defined to form a basis for the subspaces S_q. Two methods are suggested: For the parent method, new sampling points are selected by processing the shape tree towards its root. On the other hand, the neighbor method adds new sampling points by processing the adjacent nodes of the same depth d.

A problem occurs if the basis for S_q is ill-conditioned, leading to a large sensitivity of $l_{\perp q}$ depending on the sampling points that span S_q. This effect happens either accidentally if two sampling points close together or if the trajectory is degenerated and lies in a lower dimensional subspace. To consider the effect, the condition number based on a SVD of the basis vectors of S_q is calculated. As it is unknown whether a basis is ill-conditioned by accident or due to degeneracy, the PCA eigenvalues λ_e, $e = 1, \ldots, n$ of all sampling points of the trajectory segment are calculated as well. All values - subspace projection length, condition number and PCA eigenvalues - form the similarity measure used throughout this chapter. They are stored in a tree with identical structure as the shape tree, called curvature tree $\mathscr{C}$. The method is related to LTE but not identical.

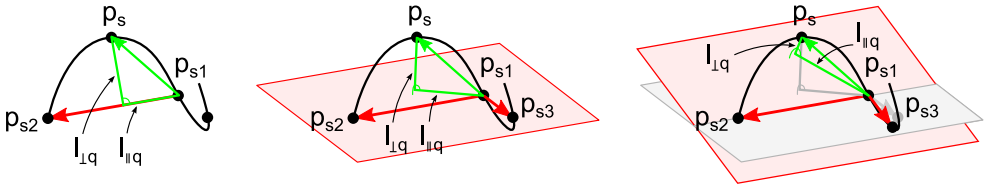


Figure 3.11: Curvature tree creation overview. Left: 2D example. Middle: 3D example. Right: Effect of an ill-conditioned basis

For equidistant and subsequent sampling points, the measure $l_{\perp 1}$ is similar to first order finite difference constraints as in (2.7) and the measure $l_{\perp 2}$ corresponds to second order finite difference constraints in (2.9). For higher orders the two approaches differ. LTE relies on a regular mesh structure, specifies only local path properties and is defined only for left/right/central finite differences. Subspace projection does not come with these limitations, yet it suffers from ill-conditioned bases.

Path comparison

This paragraph describes how the similarity measure l_f is calculated based on the values in each curvature tree. It is based upon a four-staged approach:

- Find corresponding nodes of two curvature trees
- Normalize all values of each node
- Calculate the difference between the values of two corresponding nodes
- Sum all differences to obtain l_f

The first aspect is important if two curvature trees with different structure are compared. The second aspect prevents any single effect (subspace projection length, condition number or PCA eigenvalues) to become dominant due to a degenerated path. It also makes the descriptor scale invariant. The last two aspects described how the final similarity measure is calculated based on the individual elements of the curvature tree.

To find the corresponding nodes of two curvature trees, the two trajectories to be compared are marked with superscripts ^{a} and ^{b}. When introducing a mid position $r_m \in [0,1]$ for the order of all nodes of a specific depth $d^{\{a,b\}}$, the position $c^{\{a,b\}} \in \mathscr{C}^{\{a,b\}}$ of every node in the tree is defined by the tupel $\{r_m^{\{a,b\}}, d_{r_m}^{\{a,b\}}\}$. The corresponding node c_o of the other curvature tree is found as

$$c_o = \underset{c^b \in \mathscr{C}^b}{\operatorname{argmin}}(\|r_m^a - r_m^b\|) \quad \text{s.t. } d_{r_m^a} = d_{r_m^b},$$

that is the node with similar depth that minimizes the absolute difference between the two r_m-values. To calculate the r_m-value of every node, a iterative scheme evaluates all nodes of the tree. Its algorithmic description with the start and end position $\{r_s, r_e\} \in [0,1]$, the superscript ^{root} for the root node, ^{p} for every parent node, ^{c1} for every chronologically first child node and ^{c2} for every chronologically second child node is shown in Alg. 7. An example of Alg. 7 is given in Fig. 3.12 During the normalization process subspace

projection length, condition number and PCA are mapped to a range of values $[0,1]$. When normalizing the subspace length, the orthogonal offset $l_{\perp q}$ is multiplied by $2/l_s^{\{a,b\}}$ as

$$\begin{aligned} l_s^{\{a,b\}} &= \|\mathbf{p}_s - \mathbf{p}_{s1}\| + \|\mathbf{p}_{s2} - \mathbf{p}_s\|, \\ l_{\perp q}^{\{a',b'\}} &= \frac{2 l_{\perp q}^{\{a,b\}}}{l_s^{\{a,b\}}}, \quad q = 2, \ldots, m. \end{aligned} \tag{3.40}$$

Algorithm 7: NodeIndex($\mathscr{C}$)

foreach $c \in \mathscr{C}$ **do**
  if $c = c^{root}$ **then**
    $r^{root}_{\{s,m,e\}} = \{0, 0.5, 1\}$

foreach $c \in \mathscr{C}$ **do**
  if *number of children* $= 1$ **then**
    $r^{c1}_{\{s,m,e\}} = r^{p}_{\{s,m,e\}}$
  else if *number of children* $= 2$ **then**
    $r^{c1}_{s} = r^{p}_{s}$
    $r^{c1}_{m} = (r^{p}_{s} + r^{p}_{m})/2$
    $r^{c1}_{e} = r^{p}_{m}$
    $r^{c2}_{s} = r^{p}_{m}$
    $r^{c2}_{m} = (r^{p}_{m} + r^{p}_{e})/2$
    $r^{c2}_{e} = r^{p}_{e}$

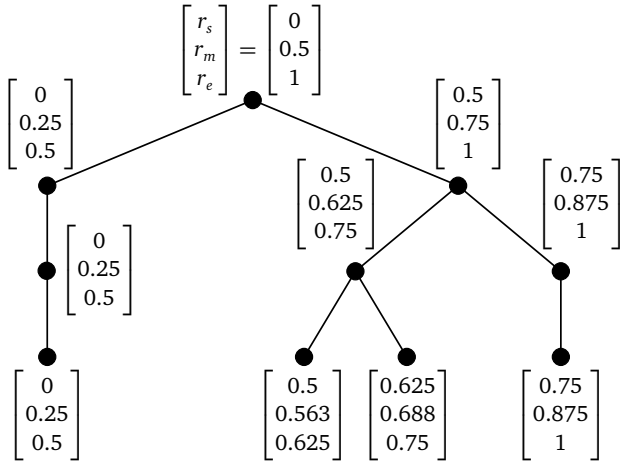


Figure 3.12: Relative position of each curvature tree node

To normalize the condition number (which is always ≥ 1), the inverse is taken as

$$\kappa_q^{\{a',b'\}} = \frac{1}{\kappa_q^{\{a,b\}}}. \tag{3.41}$$

All PCA values for a single node are mapped to the $[0,1]$ range by dividing them with the largest eigenvalue as

$$\lambda_f^{\{a',b'\}} = \frac{\lambda_f^{\{a,b\}}}{\max(\lambda_f^{\{a,b\}})}, \quad f = 1, \ldots, m. \tag{3.42}$$

For the final similarity measure l_f, the difference between the values of the two paths is taken. An additional normalization by 2^d achieves a similar influence of all the nodes of

every depth d as there is usually 1 node at $d=1$, 2 nodes at $d=2$, 4 nodes at $d=3$, etc. Once calculated, all values of all layers are summed up as

$$
\begin{aligned}
l'_{d1} &= \frac{\sum_q (|l^{a'}_{\perp q} - l^{b'}_{\perp q}|)}{2^d}, \\
l'_{d2} &= \frac{\sum_q (|\kappa^{a'}_q - \kappa^{b'}_q|)}{2^d}, \\
l'_{d3} &= \frac{\sum_f (|\lambda^{a'}_f - \lambda^{b'}_f|)}{2^d}, \\
l_f &= \sum_{\forall c^a \in \mathscr{C}^a} (l_{d1'} + l_{d2'} + l_{d3'}).
\end{aligned}
$$

to obtain the final similarity measure l_f.

Invariance properties

Theorem 5 The similarity measure l_f is invariant to rigid transformations and scaling.

Proof. As a rigid transformation preserves the distance between every pair of points, it does neither influence the splitting process nor the similarity measure. To proof scaling invariance, the coordinates of every sampling point of the trajectory are multiplied by a scalar γ_s. Then (3.37) becomes

$$
\begin{aligned}
L_{1i} &= m_L \left(\sum_{k=1}^{i-1} \left\| \gamma_s \mathbf{t}_{k+1} - \gamma_s \mathbf{t}_k, \right\| \right)^2, \\
S_{1i} &= \sum_{k=1}^{i} \left\| \gamma_s \mathbf{t}_k - \gamma_s \mathbf{t}_k^{proj} \right\|^2,
\end{aligned}
$$

In a similar way, (3.38) is rewritten as

$$
\mathbf{p}_s = \gamma_s^2 \underset{p_i \in \mathbf{P}_{1n}}{\operatorname{argmin}} (L_{1i} + S_{1i} + L_{in} + S_{in}), \quad i = 1, \ldots, p.
$$

and thus independent of the scaling factor γ_s. Neither is the final similarity measure l_f influenced by γ_s due to the normalization in (3.40)-(3.42).

■

3.9 Experimental evaluation

This section evaluates approaches presented in this chapter in real life on different robotic platforms like an HRP-4 humanoid robot and a planar 3DOF manipulator.

3.9.1 Clap/lift-the-box scenario using a HRP-4 humanoid robot

To validate the distance mesh approach presented in Sec. 3.6, two scenarios with five demonstrations per scenario are recorded. 38 markers are attached to the human and tracked by a motion capture system from Motion Analysis Inc. at $200Hz$. The whole-body free space motion is then onto a HRP-4 humanoid robot, providing the link-link distances and overcoming the problem of different link lengths between human and robot. Additional link-object distances are measured directly from the human demonstration, causing some inevitable modeling errors as the link-link and link-object distances are not perfectly matching anymore. Whereas simulations are performed in OpenRave [68], real life experiments are conducted on the HRP-4 robot. The prioritized IK method presented in Sec. 3.4.1 is used to calculate a physically feasible whole-body motion with its parameters given in Tab. 3.4.

In the first scenario - named clap - a standing human puts his hands first on the lap and claps afterward with both hands risen up. No objects in the environment are considered, thus the distance-mesh calculation is based solely on the link-link distances. This way the free space imitation capabilities of the approach are investigated. For the second scenario - named box - a box has to be lifted from the right to the left side of a table. In this scenario also the link-object distances between human and box are taken into account. The second scenario requires more precise movements than the first scenario as an unilateral contact between hands and box has to be maintained to prevent the box from falling down. Key frames of both scenarios are shown in Fig.3.13, both for the human demonstration and the robot motion reproduction. As the robot is entirely position controlled, motors and gears are easily damaged due to the closed kinematic chain when holding the box. Thus a smaller box with added compliant elements is used for safe task execution. A

parameter	value
π_d	0.03
K_{com}	0.3
K_f	0.3
K_{ca}	0.5
K_{im}	0.2

Table 3.4: Parameters for the clap/lift-the-box scenario

comparison with a conventional method based on a joint angle encoding is shown in Fig. 3.14. The figure is both split up into movement reproduction for a standing robot (top) and a sitting robot (bottom) as well as into the distance mesh based method (left) and the joint angle approach (right). Similar to the distance mesh approach the joint angle approach calculates a reference motion by using the combination of a CHMM and GMR, resulting in a time varying reference joint angle vector $\hat{\mathbf{q}}_t$. To ensure collision avoidance and a stable stand while keeping the joint error $\hat{\mathbf{q}}_t - \mathbf{q}_t$ between reference motion and reproduced motion small, a prioritized IK approach of the form

$$\dot{\mathbf{q}} = \mathbf{J}_1^{\#}\dot{\mathbf{x}}_1 + K_\theta(\mathbf{I} - \mathbf{J}_1^{\#}\mathbf{J}_1)(\hat{\mathbf{q}}_t - \mathbf{q}_t),$$

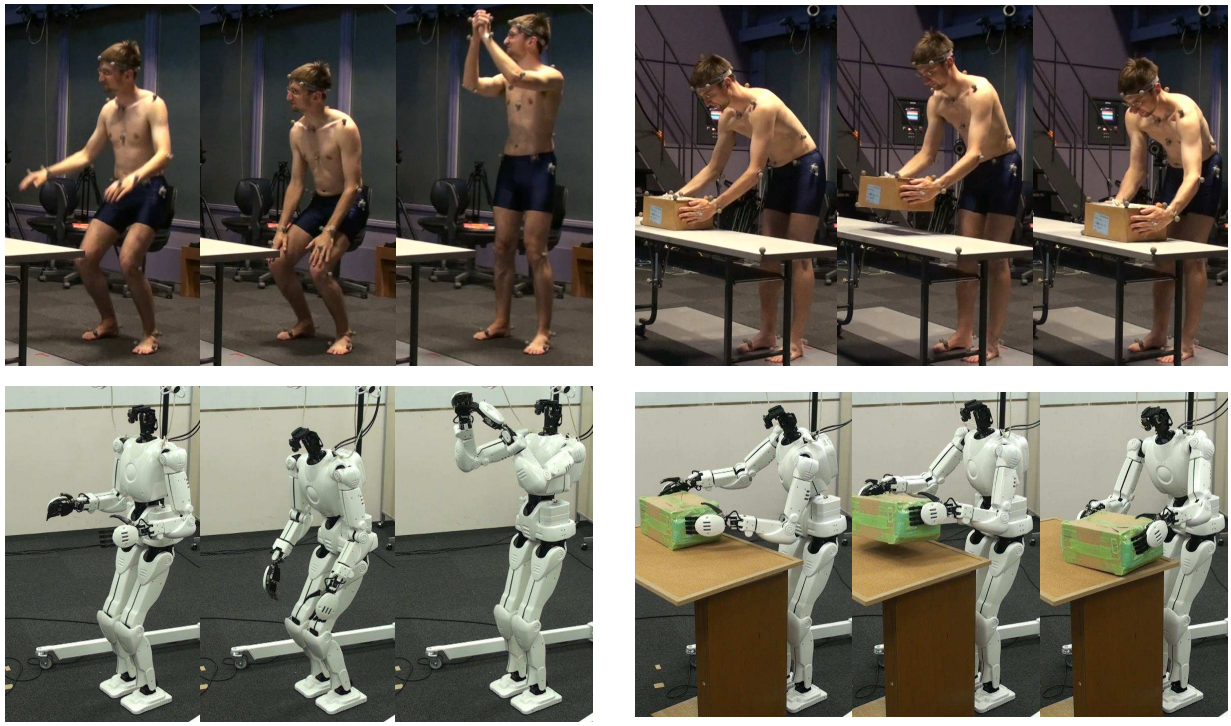


Figure 3.13: Human demonstration and HRP4 robot imitation for clap and box scenario

with $\mathbf{J}_1$ and $\dot{\mathbf{r}}_1$ similar to Sec. 3.4.1. Due to the different control laws the reproduced movements for the two scenarios differ. The difference becomes obvious for the sitting clap scenario. When using the distance mesh approach the robot tends to lean backwards as it tries to maintain the original link-link distance between legs and upper body. The joint angle approach fails to place the robot's hands on the lap because it considers only joint angles but ignores the spatial relation between lap and hands. Two additional plots on the bottom illustrate the imitation quality both in terms of the imitation cost function c_{cf} and the summed joint angle error $\sum_{i=1}^{k}|\hat{\mathbf{q}}_t - \mathbf{q}_t|$. Whereas the distance mesh method provides better results in terms of c_{cf}, the joint angle approach results in a smaller summed joint angle error.

The distance mesh redundancy is evaluated in Fig. 3.15. To reduce the number of elements of the distance mesh and accelerate calculations, the two methods for complexity reduction described in Sec. 3.6 are evaluated. $n = 32$ points are used to construct the distance mesh for the box scenario, resulting in a maximum of $(32 \times 31)/2 = 496$ elements for the distance mesh. In contrast the clap scenario consists of only $n = 21$ points, corresponding to a maximal number of 210 distance mesh elements. Based on a given reference reproduction using all elements of the distance mesh the average task space error of all the points $\mathbf{p}_{i,t}$ summed up over all time steps is shown. It is visible how a reduced number of task space elements leads to a larger average error, corresponding to a worse imitation quality. For both scenarios there is a defined threshold where the average error suddenly increases and the robot fails to imitate the desired movement. Fig. 3.16 illustrates the effect of the modeling/measurement error handling scheme. Two different

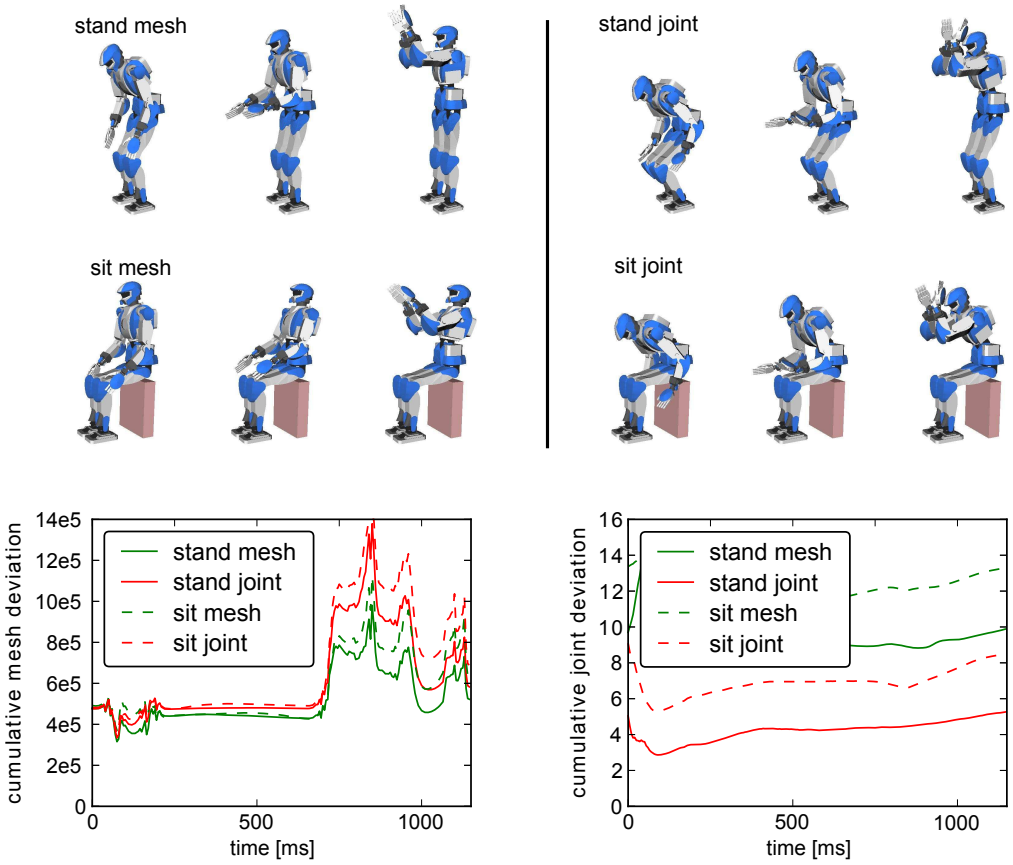


Figure 3.14: Comparison between the distance mesh approach and a joint angle approach for a standing and sitting robot. Left: Distance mesh approach. Right: Joint angle approach. Two plots on the bottom show the value of the imitation cost function c_{cf} and the summed joint angle error

box scenarios with the weighting matrix $\mathbf{W}_t$ chosen either as $\mathbf{D}_{man}\hat{\mathbf{\Sigma}}_{s,t}^{-1}$ (with manipulability) or $\hat{\mathbf{\Sigma}}_{s,t}^{-1}$ (without manipulability) are compared. Differing from the second scenario the first scenario does not consider any COM, feet or collision constraints. Shown is the absolute velocity of the fastest moving joint versus execution time. If the robot enters a singular configuration during execution caused by a modeling/measurement error, it is detected indirectly through a large joint velocity. Such configurations exist a 250ms and 2100ms. Yet as the method is encoded in the secondary, lower prioritized task, its effect decreases with increasing number of constraints in the higher prioritized task as shown on the right side. The distance mesh approach shows that task-specific properties can be extracted automatically by a solely task space dependent measure. This makes a weighting scheme between possibly concurrent objectives in joint space and task space obsolete. Up to a certain extent it also verifies the applicability to precise manipulation tasks including closed kinematic chains. A problem of all variance-based methods is the incorporation of hard positional constraints. Due to measurement noise, a probabilistic encoding and a pseudoinverse-based differential IK such constraints are never fulfilled

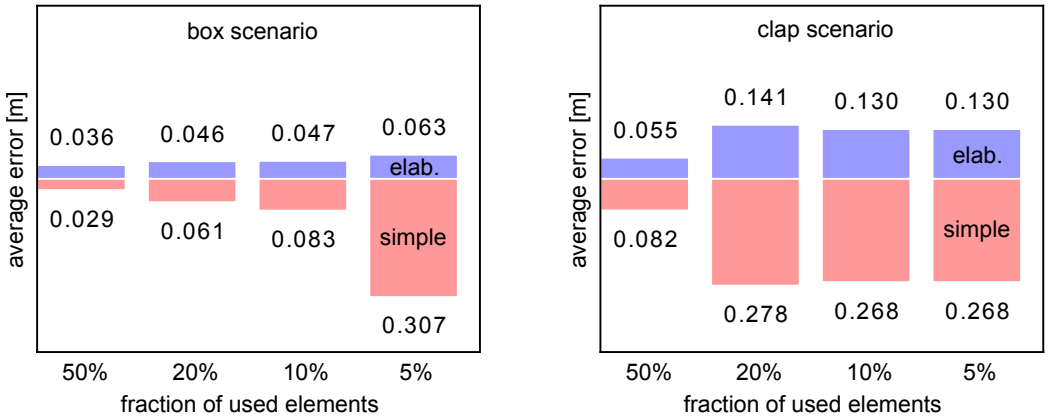


Figure 3.15: Average positional error of all distance mesh points depending on the relative number of used distance mesh elements for both scenarios

perfectly, requiring some compliance of the objects to interact with. The method is robust towards changes of the parameters $K_{com}, K_f, K_{ca}, K_{im}$ as all of them can be varied by more than one magnitude without substantial changes of the reproduced motion. It is however sensitive to parameter changes of the CHMM/GMR, especially the number of used states. Another challenge is the automatic extraction of task features based on the variance matrix $\hat{\mathbf{\Sigma}}_{s,t}$ as a small variance can be caused either by a highly task related aspect or a completely unrelated aspect that does not change the task space distance during demonstration.

3.9.2 Obstacle avoidance scenario using a HRP-4 humanoid robot

A small obstacle avoidance scenario validates the PL system approach in Sec. 3.7. The challenge consists of a button to be pressed in the presence of obstacles. Five human demonstrations are recorded in which the hand is moved from various starting positions through a narrow tunnel to press the button, see Fig. 3.17. The movement is then adapted with a HRP-4 robotic platform for which the used IK parameters are given in Tab. 3.5. To

parameter	value
π_d	0.001
K_{com}	0.3
K_f	0.05
K_{ca}	0.5
K_{im}	1

Table 3.5: Parameters for the obstacle avoidance scenario

adapt the motion to a different environmental situation and safely avoid obstacle while maintaining the task-specific properties, the approach is combined with DTW [148] and GMR [46]. DTW aligns all demonstrated trajectories in the temporal domain. The result is used by GMR to give a prototypic trajectory with corresponding spatial variance $\mathbf{\Sigma}_i$

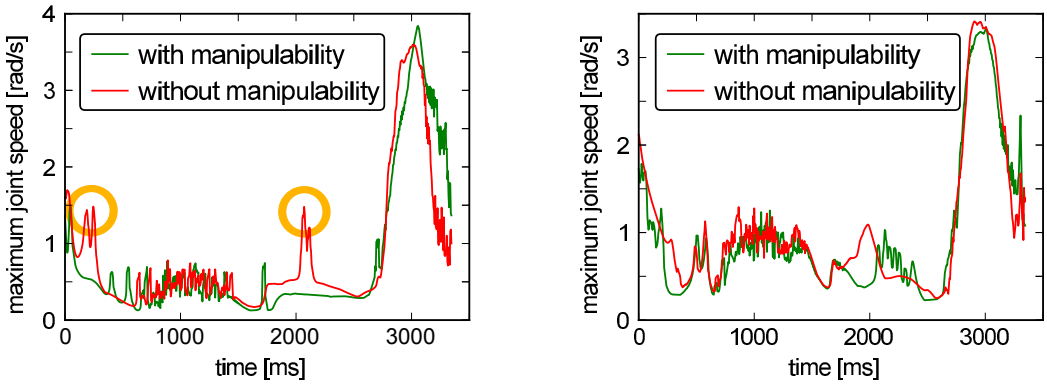


Figure 3.16: Effect of the modeling/measurement error handling scheme. Left: Velocity of the fastest moving joint versus execution time with reduced number of constraints. Right: Speed of the fastest moving joint versus execution time when considering all constraints. Close to singular configurations are highlighted with orange circles

for every sampling point $\mathbf{p}_i$. The strength parameter κ_i in (3.7) depends on the spatial variance along the trajectory as

$$\kappa_i \propto \frac{1}{\sqrt{\det(\Sigma_i)}},$$

This ensures quick convergence and small deviations from the prototypic trajectory whenever the spatial variance is small, e.g. in heavily constrained environments. At the same time it also allows a greater flexibility if the spatial variance is large, corresponding to free space motion. This effect is illustrated in Fig. 3.17. The left side represents an abstract view of the task. Grey boxes mark the two obstacles, a green and red cylinder stands for the buttons to be pressed. The five human demonstrations are visualized by orange trajectories. The colorful tube is centered around the prototypic trajectory. Whereas the tube color represents the variable PL system strength κ_i, the tube diameter is directly proportional to the distance $d_{m,s}$ of the spatial bounds approximation in (3.36). It is visible how the small spatial variance of the demonstrated trajectories between the obstacles result in a large κ_i and ensure quick convergence. On the other the large spatial variance at the beginning of each trajectory due to the different start positions results in a small κ_i value. The right side shows the adaptation to a different environment where both obstacle position and goal position, i.e. the button to be pressed, are changed. By deforming the trajectory through LTE, collision avoidance according to the spatial bounds approximation is guaranteed within the tube. The two bottom rows of Fig. 3.17 illustrate the adapted motion for two different start positions Fig. 3.18 shows the schematic view of the obstacle avoidance scenario with the different convergence strength depending on the spatial variance.

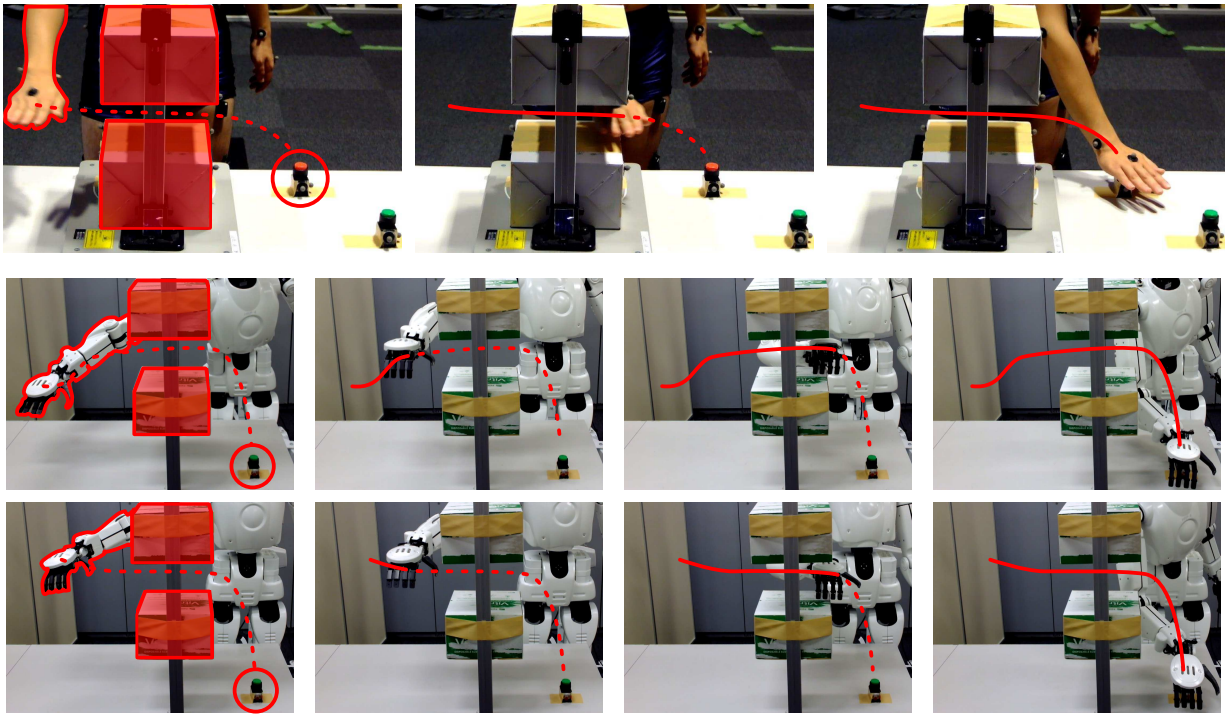


Figure 3.17: Obstacle avoidance scenario. Top: Demonstrated human motion. Bottom: Adapted robotic motion

3.9.3 Model-free motion classification

Two experiments are performed to evaluate the classification scheme in Sec. 3.8. In the first experiment the classification rate is evaluated using a proprietary data set of recorded trajectories. The second experiment compares the proposed method with other learning-free methods using the Australian Sign Language dataset [136].

Classification of human free space motion

The first set of experiments classifies a set of 3D human hand gestures tracked by a Kinect camera. The data set consists of 40 different movement classes with 5 samples per class, see Fig. 3.19. 32 different gesture form the 32 base movement classes. The other 8 classes follow a similar path as 8 corresponding base classes but the movement is stopped at a certain point of the trajectory while continuing to recording data. This evaluates the classification rate of similar paths with a different spatial distribution of all sampling points along the path. The 5 samples of each class vary in orientation, speed and scaling. To suppress measurement noise, the measured data is preprocessed using a moving-average filter with 5 frames and outlier removal for single points. The trajectories consist of 62 to 564 sampling points with images of all classes given in Fig. 3.19. All classification-relevant computations are conducted in Matlab R2010a under Ubuntu 10.04LTS using an Athlon Phenom II X6 1075T CPU with 8 GB RAM. Results when comparing the 32 base classes are listed in Tab. 3.6. The percentage of correct classification is on a 1:1

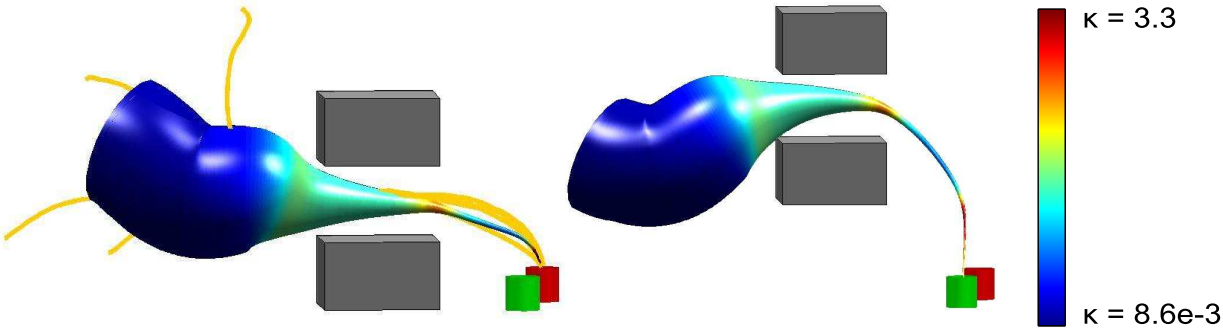


Figure 3.18: Schematic view of the obstacle avoidance scenario. Left: Original scenario with demonstrated trajectories (orange) convergence properties towards the prototypic trajectory (colorful tube) and convergence speed (color of the tube). Right: Adapted scenario with changed obstacle position (grey cubes) and different goal position

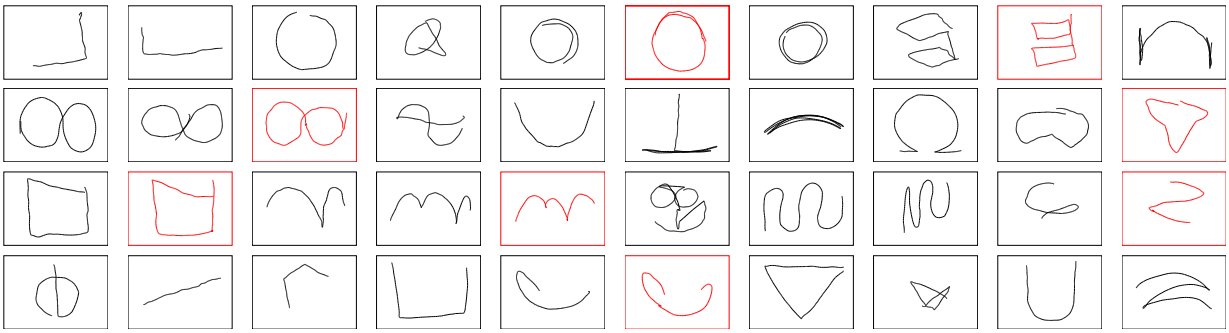


Figure 3.19: Overview of the 40 different 3D classes for trajectory classification. Every black trajectory belongs to one of the 32 “base’ ’classes. Each one of the other 8 red trajectory classes is a variation of one base class with paused movement at a specific point along the trajectory

basis, checking individually for every trajectory whether the trajectory with the highest similarity belongs to the same class. From left to right every row denotes the maximum compared depth d of the curvature tree, the percentage of correct classifications for the parent and neighbor method, the total time required to compare each trajectory with each other and the average time for comparing a single trajectory with another single trajectory. The last one equals the total time divided by the number of possible combinations, that is $(32 \times 5)^2$. It is good to see how the classification rate saturates for a maximum depth $d = 3$ with only small improvements for higher depths, indicating that most of the trajectory-specific features are not encoded on a local but rather global scale for the given dataset. Classification results for all 40 classes are shown in Tab. 3.7. Compared to Tab. 3.6 a decrease in the correct classification rate of around 7% is observed. The third experiment shows classification results for a set of 160 six-dimensional (6D) trajectories. To obtain the 6D trajectories from the data set of 3D trajectories, trajectories from two different classes are interpolated for a similar number of sampling points

proc. d	parent	neighbor	time [s]	time 1:1 [us]
5	93.8%	90.6%	16.6	648
4	92.5%	91.3%	8.36	326
3	89.4%	89.4%	4.33	169
2	80.6%	80.6%	2.32	90.6
1	45.0%	53.1%	1.33	51.9

Table 3.6: Classification experiment 1: 3D trajectories, 32 classes, no noise

proc. d	parent	neighbor	time [s]	time 1:1 [us]
5	86.5%	87.0%	25.8	645
4	86.0%	86.5%	13.5	338
3	81.5%	83.0%	7.00	175
2	73.5%	73.0%	3.58	89.5
1	37.5%	45.5%	2.04	51.0

Table 3.7: Classification experiment 2: 3D trajectories, 40 classes, no noise

and concatenated to form a 6D trajectory. Results in Tab. 3.8 show a decrease in the correct classification rate of around 4% when compared to Tab. 3.6. Tab. 3.9 displays

proc. d	parent	neighbor	time [s]	time 1:1 [us]
5	90.0%	90.6%	19.0	742
4	90.0%	90.0%	10.1	394
3	84.4%	87.5%	4.40	172
2	74.4%	77.5%	2.53	98.8
1	44.4%	48.8%	1.56	60.9

Table 3.8: Classification experiment 3: 6D trajectories, 32 classes, no noise

classification results for the fourth experiment with 160 3D trajectories. Different from all previous examples Gaussian noise $\mathcal{N}(0, 0.002)$ is added to every sampling point independently along every dimension, see Fig. 3.20. Compared to the first experiment in Tab. 3.9 the correct classification rate drops by around 10%. The low drop sounds reasonable as the main differences between the different trajectories are encoded in the local features but the added noise mainly affects local features. The experiments show high classification rates of the proposed classifiers in the $\approx$ 90% range under various conditions. As the calculation of the curvature tree is independent of the classification process, a computationally efficient method is to calculate the curvature tree once for each trajectory a priori and store it for further comparison with other trajectories. The classifier does not depend on any tunable parameter and can thus be quickly applied to generic problems without any parameter tuning.

Figure 3.20: Examples of noisy trajectories for the fourth classification experiment

proc. d	parent	neighbor	time [s]	time 1:1 [us]
5	83.1%	80.6%	16.0	625
4	85.6%	85.6%	8.87	346
3	80.6%	80.6%	4.39	171
2	65.0%	68.8%	2.46	96.1
1	36.9%	40.6%	1.40	54.7

Table 3.9: Classification experiment 3: 3D trajectories, 32 classes, with noise

Classification of the Australian Sign Language data set

To compare the proposed method with other learning-free classification algorithms, a common classification test is performed on the Australian Sign Language (ASL) data set [136]. Similar to Croitoru [60], Vlachos [306] and - with reservation, as their classes are slightly different - Keogh [148], a set of 10 classes ("Norway", "cold", "crazy", "eat", "forget", "happy", "innocent", "later", "lose", "spend") with 5 samples per sign is processed. For every of the 45 possible class pairings, the 10 samples are clustered through group-average agglomerative clustering creating a dendrogram depending on the similarity between the two sequences. If the highest level of the dendrogram separates all samples of one class from all samples from the other class, it is considered as correct classification. The percentage of correct classification rates is shown in Tab. 3.10. For a processing depth $d = 3$ the success rate of the proposed method is comparable with other model-free approaches, for a processing depth $d \geq 4$ existing comparable approaches are outperformed. Compared to the 1:1 classification for the human free space motion the

proc. d	parent	neighbor	Croitoru	Vlachos	(Keogh)
5	62.2%	62.2%			
4	60.0%	57.8%			
3	51.1%	51.1%	53.1%	46.6%	51.1%
2	33.3%	37.8%			
1	2.2%	2.2%			

Table 3.10: Classification experiment 5: ASL data set

dendrogram-based classification is a much harder problem, leading to lower classification rates. If only a single trajectory is classified wrongly, the entire classification for the two classes to be compared fails.

3.10 Summary

The LTE planning framework is expanded in this chapter to account for robotic control applications. Different control schemes are developed which - when combined with LTE - overcome specific problems of LTE associated with online control. The chapter follows two main ideas, expanding the trajectory-driven nature of LTE to whole-body control schemes and accounting for the local trajectory properties on a controller level. To consider dynamic constraints, upper bounds are derived which link the trajectory deformation in task space to the applicable torque limits. Despite their generality, some methods are computationally demanding as they require a LTE update of the reference trajectory at every time step during execution. Computational complexity is further increased by using a predictive controller to account for the interlocking nature of local trajectory properties.

3.11 Bibliographical notes

The results in this chapter and the related appendix are either partly or in their entity based on the author's work in [224, 227, 229]. All the singular-robust IK approaches in the style of (3.3), (3.4) and (3.12) tackle the general problem of regularizing an ill-posed matrix and are better known as Tikhonov regularization [300]. While working on the task space distance mesh approach, a conceptually similar method is presented in [120, 218] to control the motion of a humanoid robot/character. Although overlapping at certain points, e.g. when using a singular-robust IK approach to create task-specific motions, the detailed implementation and scope differs and is thus difficult to compare. Wheras the HMM-based task space distance mesh is primarily used to automatically extract the task-related features, their work focuses on movement adaptation and dynamic balance constraints. As mentioned in the text, the method of iteratively subdividing a trajectory and the word choice of "shape tree" have been used previously in [86].

4

Motion reasoning

This chapter investigates how to decompose a general task through motion reasoning into single motions to process them efficiently in the context of motion imitation. Existing state-of-the-art approaches in PbD focus solely on specific motions but not their interconnection to achieve an abstract task. In the latter case the challenge of motion reasoning becomes to find automatically both a set of possible motions and suitable constraints for an optimal task completion. By predicting the outcome of each motion for a given situation and simulating several motions in advance, an optimal motion is selected. As the approach is model-based, it has to resemble the environment precisely. For cooperative or competitive scenarios involving humans, their behavior has to be modeled through psychological experiments. As shown in this section, this leads to a significant improvement in overall task completion rate.

The chapter is organized as follows: Sec. 4.4 introduces the general framework for motion reasoning based on an MDP representation. Application-specific adaptation mechanisms are described in Sec. 4.5, both in the context of how an desired action can be represented and how the general approach has to be adapted to fit task-specific requirements. Sec. 4.6 is dedicated to derive a suitable, yet general model of inaccuracies arising at a game of pool. All aspects are combined in the general framework and evaluated both through simulations and experiments with an anthropomorphic platform in Sec. 4.7. Whereas Sec. 4.8 provides a summary about the motion reasoning framework and its adaptation to pool, Sec. 4.9 lists existing literature on this topic.

4.1 Introduction

Under the prerequisite that both a reference motion and constraints are given, LTE allows for a proper motion adaptation while taking into account specific properties of the

reference motion. Whereas the last two chapters showed a variety of options how to generate a reference motion, it is still mostly unknown how to derive suitable boundary constraints for a complex task pursuing a long-term goal. When specified manually, the human has to be an expert for the given task as wrong constraints may have either no or a counter-productive effect. Another option is an automatic extraction of task-specific constraints from multiple demonstrations, yet this method assumes that the task is encoded in the difference between the demonstrations. This is a highly specific assumption and only of limited scope for the general case.

In many scenarios the task is not described by a single motion but a multiplicity of motions. Looking just at a single motion provides no insights about the overall task, rather it is the interaction and coupling between the different motions. Take soccer as an example: Just shooting at the goal from any position on the field is probably the simplest way to score a point but by far not the most efficient one. A better way is to rely on combinations and tactical play consisting of several passes until finally a player stands free and can make a close goal shot. In such a situation it is necessary to find an optimal motion for a long-term goal (shooting a goal) rather than just the instantaneous situation (make a pass). To find the best action possible, it is necessary to specify the effect of each action on the long-term goal in a quantitative manner [15, 19, 297]. Depending on the goal this becomes a strenuous task as the effect might be delayed or very subtle. A straightforward method is to rely on expert knowledge about the task. If there is no expert at hand, it can be also learned from previous experience.

All methods lead to a model-based approach where one expects a specific outcome for each action [83]. This makes it possible to plan ahead, e.g. for a given situation S_1 an action A_1 is found which will result in a new situation S_2 etc. By simulating all situations and actions one finds an optimal action for each situation. The more precise the model is, the better it will represent the true situation and circumstances and the more realistic it will be. If the task relies on or involves the interaction with humans, their behavior has to be included in the model as well. This is a challenging duty as in many cases the involved person is neither aware of its intention nor the outcome of its action. Thus psychological experiments have to model both the human preference and effect accordingly.

4.2 Related work

Motion reasoning as described in the introduction is closely related to motion interaction and prediction - in order to reason about a suitable motion one has to be aware of its outcome and possible consequences. Various modi of interactions are possible [63]. A different style of interaction can be achieved either on a physical basis [123, 128, 168, 265] or using gestures, visual information and social interaction [85, 146, 167, 221]. The same accounts for the level of interaction [104], ranging from learning-based agents [126, 200] enabling them to predict future actions of the interaction partner to the intention of a participating subject, going beyond the set of observable action towards figuring out the underlying action goals [66], either in collaborative scenarios [32, 81, 231, 266] or competitive scenarios [36, 310].

Competitive games are an appealing test bed to evaluate the motion reasoning capabilities [52, 305]. As game rules are well defined, the evaluation of motion reasoning is easy - if you win, you're better than your opponent. Still they require various level of skill in terms of motion control, motion planning and motion reasoning. Depending on the game, focus can be on any of the three - whereas tug of war is more about motion control, chess depends on superior motion reasoning capabilities. Motion reasoning in the context of motion imitation and adaptation becomes especially important if there is no unique focus on any of the three but all are essential for a successful task completion. Due to its multi-faceted aspects combining all aspects, billiards (from now on called "pool") is used as a representative scenario in this chapter. Players need a certain level of motion control in order to pocket balls reliably but also have to reason about future actions for a good position play.

To master the game of pool, different factors have to be considered. A pool simulator is required for a model of the environment, e.g. PoolFiz [183] which enables one to plan several motions (strokes) ahead. Different methods exist when reasoning about the optimal next motion, either through a densely sampled search tree [13, 276], gradient-based optimization [176, 177] or fuzzy logic [176, 192]. By treating pool from a game-theoretic perspective, it was also found to feature the existence of Nash equilibria [14]. Several pool robots have been developed within the last 20 years [9, 10, 53, 54, 105, 296], yet they used either highly specialized gantry-based robots or use modified cues that fit the robot.

4.3 Problem statement and conceptual approach

Referring to the original problem formulation (1.1) at the beginning of the thesis, motion reasoning is about finding suitable boundary constraints $h(u)$ for an abstract task u that is processed by an underlying motion adaptation scheme.

$$\begin{aligned} &\text{optimize } g(x, \hat{x}), \\ &\text{s.t. } h(u). \end{aligned}$$

The constraints $h(u)$ must be chosen such that the movement is fulfilled and the task will be executed in an optimal manner. Thus the motion reasoning scheme has to be both be aware of which constraints lead to executable movements and how they are derived in an optimal manner. During motion reasoning it is assumed that only one action (movement) $A \in \mathscr{A}$ happens at each state (situation) $S \in \mathscr{S}$, leading to a new state. For every state it is then calculated which action to perform next and how to define the constraints until the task is finally achieved. If the Markov property holds, i.e. each action depends only on its single prior state, the task can be modeled as a MDP [25]. There exist effective solution algorithms for MDPs [30, 44], calculating an optimal action for each state while allowing also for the outcome of subsequent actions. As the simulation of subsequent actions depends on a precise model of the environment, the major challenge of this approach is to find a holistic representation covering all task-relevant aspects. Two major aspects must be considered when modeling the environment besides the task itself,

differing both in their representation and the effect on each action. They are limitations and inaccuracies. Although there are exceptions, limitations are generally treated in a deterministic manner. Various types of limitations exist, including but not limited to kinematic constraints, dynamic constraints, safety aspects or artificial constraints for a pleasant interaction. Yet their effect is the same, reducing the set of all possible actions $\mathscr{A}$ to a subset of actions $\mathscr{A}_{red}$ that does not violate the limitation constraints as

$$\mathscr{A}_{red} \subset \mathscr{A}.$$

Inaccuracies on the other hand are usually treated in a probabilistic manner as their effect is best predicted in a probabilistic fashion. Due to limited precision, large tolerances, measurement or sensing errors or other hidden effects the executed action $\tilde{A}$ differs from the planned action by the unknown function $f(A)$ as

$$\tilde{A} = f(A).$$

Without inaccuracies, $f(A)$ would be the identity function and the executed action is similar to the planned action. When considering inaccuracies, $f(A)$ is a probability, yet unknown function. Both limitations and inaccuracies must be considered if the model has to resemble the real world properly. Throughout the chapter the game of pool is used as a representative scenario with a robot facing a human opponent. During the game, limitations and inaccuracies occur in different context both as an undesired, parasitic effect but also as a tool to represent the skill of each player. Three main factors are identified:

- Limitations: Kinematic constraints
- Inaccuracies: Hardware inaccuracies and a unknown human game style

Whereas the first two aspects refer to the limited motor skill of the robot, the last is solely human-related. Kinematic constraints are caused by the joint angle limits and the workspace constraints of the robot, preventing it to bend over the table and to reach all positions. This aspect only limits the robot - the human's intrinsic motor skill is far more advanced so this hardly constitutes any impairment to the human player. Hardware inaccuracies come from a finite sensor resolution and an imperfect mechanical setup, leading to a nondeterministic stroke outcome of the robot. Whereas the human's stroke outcome is also nondeterministic, the reason is most likely due to a specific perspective and limited experience. The last aspect is the human game style: As the robot interacts indirectly through the game of pool with the human, it has to be aware of its specific game style. Differing from a machine which is supposed to act rational, humans have individual preferences which might detain them from doing the (according to robot preferences) optimal stroke. This knowledge helps the robot to improve its own game play by counterattacking the human preferences, thus it has to be modeled through psychological experiments. Out of the three aspects, the unknown human game style is by far the most difficult one to model and hardly tackled in literature. For competitive games it is in general assumed that every player has either perfect knowledge or at least acts rationally. Yet as this chapter will show, such an assumption is wrong for certain cases. When modeled properly,

incorporating an individual human game style significantly increases the success. Thus a large part of the chapter is dedicated to derive a proper model of the individual players' preferences.

4.4 Optimal motion reasoning using MDPs

On an abstract level pool can be represented as a sequential stochastic game with continuous action and state space. Whereas the action space is spanned by the set of all possible strokes, the state space is defined by the position of all balls on the table and the actual game situation given by the rules. It is sequential due to the turn-based stroke order and stochastic due to the non-deterministic outcome of every stroke. Under these conditions the Markov property holds. When discretizing the action and state space, pool is modeled as a MAXPROB MDP [161]. Similar to a standard MDP it is defined by tuples of the form $\left\langle \mathscr{S}, \mathscr{A}, \mathscr{T}, \mathscr{R}, \mathscr{G}, s_0 \right\rangle$ where

- $S \in \mathscr{S}$ is the set of states
- $A \in \mathscr{A}$ is the set of actions
- $T \in \mathscr{T}$ is a transition function $\mathscr{S} \times \mathscr{A} \times \mathscr{S} \rightarrow [0;1]$ denoting the probability of moving from state S_i to S_j by executing the action A
- $R \in \mathscr{R}$ is a mapping $\mathscr{S} \times \mathscr{A} \rightarrow \mathbb{R}$ specifying action rewards
- S_0 is the start state

As indicated in the introduction, both MAXPROP MDPs and conventional MDPs are described by a discrete transition between states (situations) and actions (what to do in which situation). For a given state S_i, multiple actions exist that lead to new individual states S_j. The outcome of each action is described by the transition function T, specifying with which percentage an action A will result in a state S_j for a given state S_i, i.e. the probability of each stroke outcome. In addition, an individual action reward R is assigned to each action for a given state, rating the outcome of each action. The general goal for MDPs is to find the action for a start state S_0 that maximizes the action reward - not only for the next action but for all future actions. Once calculated, this will give rules for all states specifying which action A to take when being in state S. The set of all rules is called policy π. The optimal policy π^* is the policy that maximizes the action reward over all policies π, defined as

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}^{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \right] \Bigg|_{S_0 = S} . \tag{4.1}$$

In (4.1) the expected reward $\mathbb{E}^{\pi}$ is used instead of a deterministic reward, accounting for the probabilistic nature of each action A. The argument in the square brackets is the with $\gamma \in [0,1]$-weighted expected reward $\mathbb{E}$ over all future timesteps $t = \{0, \ldots, \infty\}$. The parameter γ weights the importance of expected rewards further in the future relative

to the immediate expected reward. The policy π^* just provides information about which action to take in which state to maximize the expected reward, yet the specific value of the expected reward is still unknown. This is calculated through the value function $V : \mathscr{S} \rightarrow \mathbb{R}$ accounting for the expected reward when being in state S. In a similar manner the value function $V^\pi(s)$ for the policy π is defined as

$$V^\pi(S) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \right] \Bigg|_{S_0=S} . \tag{4.2}$$

With the two definitions in (4.1) and (4.2), the optimal policy π^* is written in a shorter form as

$$\pi^* = \underset{\pi}{\operatorname{argmax}} V^\pi,$$

that is the policy maximizing the value function V^π over all possible policies π. The value function is closely related to the action-value function $Q : \mathscr{S} \times \mathscr{A} \rightarrow \mathbb{R}$ as

$$Q^\pi(S,A) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \right] \Bigg|_{S_0=S, A_0=A} ,$$

specifying the expected reward for the policy π when being in state S and taking the action A. Whereas the value function specifies the expected reward in state S for a multiplicity of different actions A, the action-value function assumes also assumes that one is in state A but has already decided which action A to take. Similar to the value-function, the optimal policy π^* is rewritten in dependence of the action-value function as

$$\pi^* = \underset{\pi}{\operatorname{argmax}} Q^\pi,$$

maximizing the action value function Q^π over all possible policies π.

MAXPROB MDPs are a special class of MDPs based upon a Stochastic Shortest Path MDP (SSP MDP) [31] which extend the standard MDP by a set of absorbing goal states $G \in \mathscr{G}$. Such a goal state is defined as

$$\begin{aligned} T(G,A,G) &= 1 \quad \forall A \in \mathscr{A}, \\ R(G,A) &= 0 \quad \forall A \in \mathscr{A}, \end{aligned}$$

allowing only self-transitions in G and accumulating no reward. The only two conditions for SSP MDPs are that for every state S at least one *proper* policy exists that reaches a goal state with probability $P = 1$. Moreover, every *improper* policy must have a reward of $-\infty$. Both conditions correspond to a goal state being reachable from any state with $P = 1$. This is a rather restrictive condition as it only accounts for a "success" when reaching a goal state or an undefined state when following an *improper* policy. Yet it does not explicitly account for a "loss" when reaching a dead end states. This problem is overcome by MAXPROB MDPs which specifically account for dead ends [162] as their optimal policy does not maximize the expected reward but the probability of reaching a goal state, i.e. winning the game. By assigning a reward of 1 to every action that

reaches a goal state and 0 to all other actions, the optimal value function $V^*(S_0)$ reflects the probability of reaching a goal state when starting in state S_0. Under the assumption of no policy leading to an undefined state, e.g. when entering a loop, the probability of reaching a dead end is calculated as $1 - V^*(S_0)$.

4.5 Action representation

As an action at pool corresponds to the event of executing a stroke, this section investigates how a stroke is represented, how inaccuracies and limitations influence the stroke outcome and how this knowledge is translated to suitable boundary constraints that are processed by an underlying motion adaptation framework. It also explains the consequences of the used action representation on the optimal motion reasoning framework to obtain a computationally tractable solution. The development of the underlying pool simulator is discussed in App. C.

Stroke difficulty

When executing a stroke, the cue transfers an impulse p on the cue ball. Every stroke is uniquely described by its contact position and stroke angle defined by four parameters α, β, γ and θ for a given cue ball position (x_c, y_c) on the table. Yet their influence on the stroke outcome differs. Both stroke intensity p and stroke angle θ have to be considered in any case to pocket balls reliably. The variables α, β and γ are mainly used to induce spin on the cue ball for a better position play. Thanks to the minor effect of the latter three parameters they are kept constant as $\alpha = 0$, $\beta = 0$ and $\sigma = \frac{\pi}{2}$ from now on. As a result, the parameter space for each stroke is reduced from five to two dimensions by considering only p and θ. The stroke difficulty for direct strokes is calculated using basic

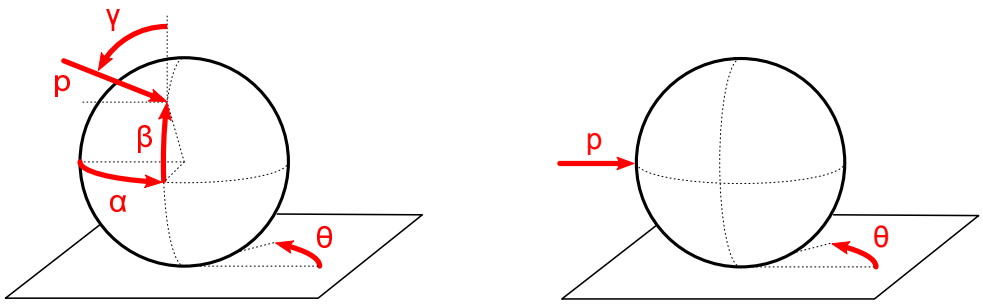


Figure 4.1: Stroke parameter overview. Left: All possible parameters. Right: Considered parameters in this thesis

geometry if the following conditions are fulfilled

- The transferred impulse in tangential direction during ball-ball collisions is negligible
- The stroke intensity p and stroke angle θ are independent of each other

- The stroke intensity p being larger than a minimum value p_l

In this case the difficulty of a stroke is specified by the range of values $\theta_l \leq \theta \leq \theta_u$. The values θ_l and θ_u mark the two limiting cases for which the object ball is just pocketed without touching the cushion. The value

$$\theta_m = \frac{\theta_l + \theta_u}{2},$$

is the mean angle for which the object ball is pocketed in the middle of the pocket. The allowed angular deviation (AAD) is defined as

$$\text{AAD} = \theta_u - \theta_m = \theta_m - \theta_l, \tag{4.3}$$

and denotes how far one can deviate from the mean angle while still pocketing a ball. The smaller the AAD is, the more difficult the stroke will be. Note that the approach can be readily extended to combination shots with more than one object ball.

Another stroke difficulty measure is based upon expert knowledge [56, 76]. It reveals that the human perception of stroke difficulty depends only on a few parameters: The distance d_1 between cue ball and object ball, the distance d_2 between object ball and pocket and the "cutting angle" θ_c, see Fig. 4.2 right side. Various approaches in literature try to deduce an expression for the difficulty of a stroke from this knowledge [192, 276], one of the most prominent ones quoted in [177] as

$$\kappa_l = \frac{cos(\theta_c)}{d_1 d_2}. \tag{4.4}$$

In a similar way to the AAD method, a small value corresponds to a tricky stroke, a large value represents an easy one. The AAD method is the correct way of measuring the difficulty for the stated assumptions from a bird's eye view, implicitly considering the parameters θ_c, d_1, d_2 and partially blocked paths caused by other balls on the table. The expert method on the other hand suggests that the human perception of difficulty deviates from the AAD method due to a different perspective, thus requiring a new difficulty measure when modeling the human game style. Hardware limitations caused by kine-

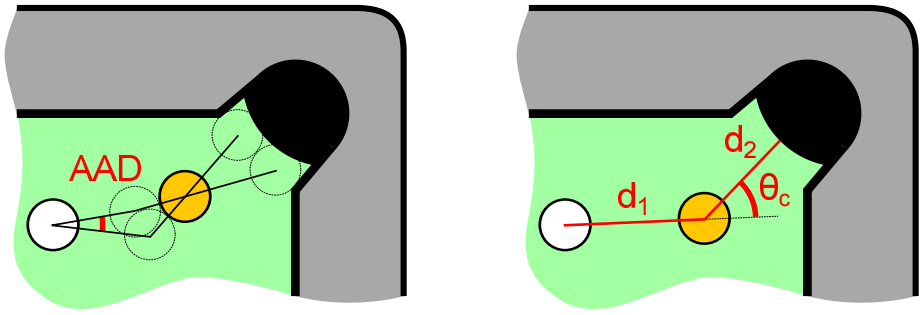


Figure 4.2: Stroke difficulty comparison. Left: AAD method. Right: Expert method

matic constraints directly affect the stroke difficulty as they limit the possible range of θ

values. The AAD method accounts for kinematic constraints in a straightforward manner as a θ limitation directly influences the range of feasible stroke values $[\theta_l; \theta_u]$. The boundaries $[\theta_l; \theta_u]$ are calculated through an iterative collision checking between robot and table. For the expert method however, kinematic constraints are only considered approximately. As such a stroke is defined impossible if θ_m is not within the possible range of θ values. Fig. 4.3 visualizes the effect. Shown is a simplified, yet true-to-scale collision model of the used pool robot and the pool table, approximated by bounding boxes. The angle $\theta_r \in [0; 2\pi]$ defines the possible stroke execution directions for a given position of the white ball on the table. An angle $\theta_r = 2\pi$ means the white ball can be shot in any direction whereas an angle $\theta_r = 0$ stands for an unreachable white ball. The robot is limited by two constraints. First, it has a stiff body which collides with the table if the white ball is placed too far away from any cushion. Second, its right EE does not lie on the table but on the cushion, which causes problems if the white ball is placed too close to the cushion.

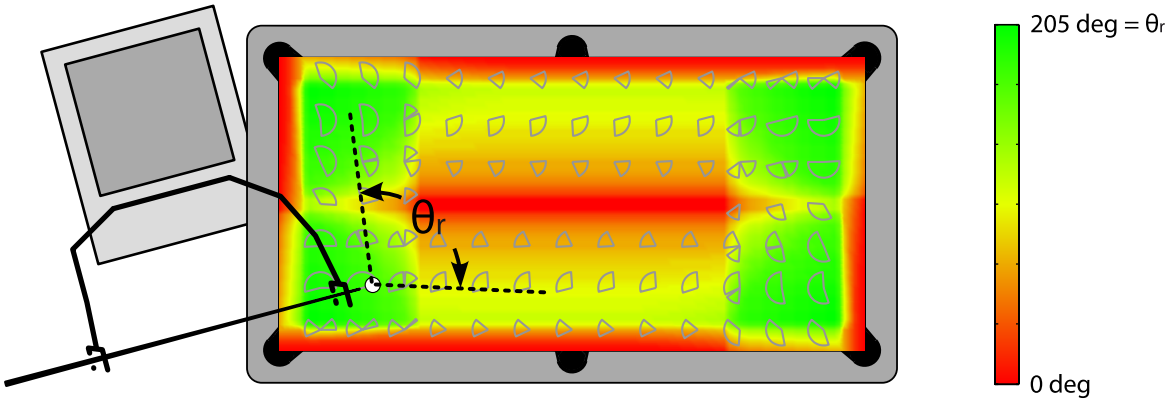


Figure 4.3: Kinematic constraints of the robotic system. Due to collisions with the pool table, the robot cannot execute a stroke in an arbitrary direction for a given ball position. The greener the area is, the larger the range of possible stroke directions is. Gray circle segments mark the possible stroke direction for certain positions of the white ball

Pocket probability

The stroke difficulty makes it possible to rank all strokes depending on their AAD. Still it is only an abstract measure, providing no information about how likely it is for a specific player to pocket a ball. This is done by the pocket probability $P_p \in [0; 1]$, connecting the stroke difficulty with the individual skill of each player to derive a percentage of success for every stroke. It is assumed that the skill of each player is represented by two normal distributions both for the stroke intensity p and the stroke angle θ as $\mathcal{N}_p(\mu_p, \sigma_p^2)$ and $\mathcal{N}_\theta(\mu_\theta, \sigma_\theta^2)$. Whereas the means μ_p, μ_θ encode the desired stroke impulse and angle, the variances $\sigma_p^2, \sigma_\theta^2$ account for the individual skill of each player. Novice players are characterized by large variances, resulting in executed strokes which deviate considerably from the planned strokes. Expert players on the other hand have variances close to zero

as they succeed in executing the planned stroke with high fidelity. By representing the effect of hardware inaccuracies through normal distributions $\mathcal{N}_p, \mathcal{N}_\sigma$, they are directly related to the skill of the robot.

The planned or desired stroke A_p is specified by the tupel (μ_p, μ_θ) as

$$A_p = (\mu_p, \mu_\theta).$$

Through the limited skill of each player, the executed stroke A will differ from the planned stroke as

$$A = f_n(\mathcal{N}_p, \mathcal{N}_\theta),$$

with $f_n(\mathcal{N}_p, \mathcal{N}_\sigma) = f_r$ as the multivariate normal distribution of $\mathcal{N}_p$ and $\mathcal{N}_\theta$.

Under the assumption of a constant minimum impulse p_l required to pocket the object ball and a constant maximum impulse p_u limited by the strength of each player, the pocket probability $P_p \in [0;1]$ is related to the AAD as

$$P_p = \int_{p_l}^{p_u} \int_{\theta_l}^{\theta_u} f_n(\mathcal{N}_p, \mathcal{N}_\theta) d\theta \, dp. \tag{4.5}$$

In general, the stroke impulse p is not limiting factor. Under the condition of $\sigma_p \ll p_u - p_l$, $p_l \ll \mu_p \ll p_u$, meaning that the standard deviation is much smaller than the range of allowed stroke impulse values and the mean stroke impulse is far away from the theoretical limits, (4.5) is approximated by

$$P_p \approx \int_{\theta_l}^{\theta_u} \mathcal{N}_\theta d\theta.$$

Then the pocket probability depends solely on the angular deviation of θ which is calculated through the stroke difficulty.

Determination of boundary constraints

The motion reasoning scheme is connected to the motion adaptation scheme through the stroke parameters. The boundary constraints $h(u)$ for the task of winning the game of pool are given by the parameters p, θ and the cue ball position (x_c, y_c), uniquely specifying each stroke. The reference motion to be adapted is specified by a number of fifth order polynomials which move the cue along a straight line. The cue is first accelerated, then kept at a constant velocity while hitting the cue ball and decelerated afterward. Whereas θ and (x_c, y_c) specify the position of the cue when hitting the cue ball, the cue velocity in the moment of hit is calculated from the desired impulse p on the cue ball. Give the boundary constraints, the motion planning framework moves the robot around the table, places the cue behind the cue ball and executes the stroke autonomously as described in [230].

Pool-specific adaptation

By representing the game of pool as a MAXPROB MDP, the otherwise continuous action and state space are discretized to be numerically tractable. This turns the probabilistic MDP in a deterministic planning problem for every state S_0. Still as both action and state space are large even for strong discretizations, heuristic search algorithms like LAO* [113] or RTDP [21] cannot be applied. Instead the true value function is approximated by evaluating only a subset of all possible state-action pairs. This gives a near-optimal value function $\hat{V}^{\pi} \leq V^{\pi}$, denoted $\hat{V}$ for simplicity. When evaluating the value function for a state S_0, a tree with S_0 as the root is created. Every vertex of the tree corresponds to a state, every edge of the tree to an instantiation of an action. As MAXPROB MDPs calculate the value function based on the percentage to reach a goal state, every branch of the tree has to be expanded until reaching a goal state or a dead end. This approach is unfeasible as there is no upper bound on the number of actions, e.g. if no ball gets pocketed.

A solution that overcomes this problem is to assign a reward not only to the action that reaches a goal or a dead end but also to other actions based on prior knowledge about the game mechanics. It is known that to win a game a player has to pocket all balls of his color. When assigning a reward $\mathscr{R}_1$ to every action as

$$\mathscr{R}_1 = \begin{cases} +1 & \text{for every pocketed ball of the player,} \\ -1 & \text{for every pocketed ball of the opponent,} \end{cases} \tag{4.6}$$

the optimal value function $\hat{V}^*(S)$ is the one for which most balls of the own color and least balls of the opponent's color are pocketed. Yet as the principal goal is still to win the game and not to pocket least balls of the opponent, the original condition of a reward of $\mathscr{R} = 1$ when winning a game becomes

$$\mathscr{R}_2 = \begin{cases} +w & \text{if player wins,} \\ -w & \text{if opponent wins,} \end{cases} \tag{4.7}$$

resulting in $\mathscr{R} = \mathscr{R}_1 \cup \mathscr{R}_2$. The effect of $\mathscr{R}_2$ becomes obvious in the endgame if one of the two players is about to win the game. Then a high w-value favors actions where the player wins the game even if some of the opponent's balls are pocketed. On the other side it also penalizes actions which give the opponent a chance to win even if the player does not pocket any ball of his own color. A parameterization of $w = 0$ and $\gamma = 1$ has a physical meaning as in this case the value function $\hat{V}(S)$ represents the expected number of balls the player will pocket more than the opponent when following the policy $\hat{\pi}$. In a similar fashion the action-value function $\hat{Q}(S,A)$ denotes the difference of pocketed balls for a given policy when executing the action A.

As a consequence it is not necessary to expand every branch of the tree until finally reaching a goal state or dead end. A relaxed approach only simulates the next n actions. The assigned reward as (4.6) and (4.7) acts as a bias towards the optimal solution. The modification leads to a new value function as

$$\hat{V}(S_0) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{n} \gamma^t R(S_t, A_{t,\{r,h\}})\right], \tag{4.8}$$

which evaluates the rewards only for $t = \{0, \ldots, n\}$.

After the search tree is created, backward induction iteratively evaluates the value function $\hat{V}(S_t)$ backward in time to find the value function $\hat{V}(S_{t-1})$. When starting with the leaf nodes S_n and processing the tree towards the root node, it will result in an optimized stroke for the actual state S_0. For the leaf nodes the backward induction step consists of finding the stroke with the highest success rate. For all other nodes first the action-value function $\hat{Q}(S_t, A_t)$ is calculated by systematically sampling the set of possible actions A_t for a given state S_t where a ball of the own color is pocketed. Such strokes are characterized according to (4.5) by

$$\begin{aligned} p_l \leq p \leq p_u, \\ \theta_l \leq \theta \leq \theta_u. \end{aligned}$$

The nondeterministic outcome of each action requires to extend the sampled action space beyond the limits $[p_l, p_u], [\theta_l, \theta_u]$ as

$$\begin{aligned} p_l - \epsilon_p \leq p \leq p_u + \epsilon_p, \quad \epsilon_p \geq 0, \\ \theta_l - \epsilon_\theta \leq \theta \leq \theta_u + \epsilon_\theta, \quad \epsilon_\theta \geq 0. \end{aligned}$$

In addition every stroke has to be weighted with its probability of occurrence, given by $f_n(\mathcal{N}_p, \mathcal{N}_\theta)$. By varying μ_p, μ_θ one finds both an optimized value function $\hat{V}(S_{t-1})$ and optimized stroke parameters μ_p^*, μ_θ^* based on the product $\hat{Q}(S_t, A_t) f_n(\mathcal{N}_p, \mathcal{N}_\sigma)$ as

$$\hat{V}(S_{t-1}) = \begin{cases} \max\limits_{\mu_p, \mu_\theta} \hat{Q}(S_t, A_t) f_n(\mathcal{N}_p, \mathcal{N}_\sigma) & \text{if player's turn,} \\ \min\limits_{\mu_p, \mu_\theta} \hat{Q}(S_t, A_t) f_n(\mathcal{N}_p, \mathcal{N}_\sigma) & \text{if opponent's turn,} \end{cases}$$

$$(\mu_p^*, \mu_\theta^*) = \begin{cases} \operatorname{argmax}\limits_{\mu_p, \mu_\theta} \hat{Q}(S_t, A_t) f_n(\mathcal{N}_p, \mathcal{N}_\sigma) & \text{if player's turn,} \\ \operatorname{argmin}\limits_{\mu_p, \mu_\theta} \hat{Q}(S_t, A_t) f_n(\mathcal{N}_p, \mathcal{N}_\sigma) & \text{if opponent's turn.} \end{cases}$$

For $\hat{V}(S_0)$ the parameters μ_p^*, μ_θ^* are the optimized stroke values for the current state. The maximization of the own reward while minimizing the opponent's reward subject to probabilistic actions is interpreted as a variant of the expectiminimax algorithm [255]. Differing from the expectiminimax algorithm it contains an optimization step for each node to calculate the optimal reward.

4.6 Human behavior modeling and countering

So far it is assumed that the stroke difficulty experienced by each player is described by the AAD according to (4.3). Yet interviews with human expert players reveal that the perceived difficulty for humans is based on a few parameters only. This includes the distance d_1 between cue ball and object ball, the distance d_2 between object ball and pocket and the cutting angle θ_c. As the measure in [177] is only determined heuristically but not verified experimentally, it is unknown which ball humans will pocket when facing multiple options. This leads to three subproblems:

- Calculating the *subjective difficulty* κ: How difficult a human thinks it is to pocket a ball.
- Calculating the *objective difficulty* P_o: How difficult it is for a human to pocket a ball.
- Calculating the human discount factor γ_h: How far a human is planning ahead.

The subjective difficulty describes the human's preference and the preferred object ball-pocket combination the player will go for. Yet unknown user preferences or limitations lead to situations where the human's preference differs from stroke with the highest pocket probability, resulting in an irrational behavior. Knowing both the human's preference and pocket probability for a given situation on the table is valuable information as it influences the expected reward in (4.8) and thus the decision of the robot. In the previous thought experiment it was assumed that the human player is not planning ahead. The amount of planning ahead is encoded in the value γ in (4.8). A value $\gamma = 0$ means that a player only considers the current state, whereas any value $0 < \gamma \; le1$ indicates that the player also values future rewards and plans ahead. For a robot this parameter can be adjusted manually. For a human player however this parameter has to be determined through experiments, resulting in γ_h. Once subjective difficulty and objective difficulty are calculated through user studies, both the human preference and pocket probability are specified in analogy to Sec. 4.5, allowing a seamless integration into the existing planning framework.

Subjective difficulty

Although the influencing factors in (4.4) are determined from expert knowledge, the function in (4.4) is specified heuristically. It is an open question whether other functions based upon the three parameters d_1, d_2, θ_c model the perceived stroke difficulty for humans better. Two experiments are performed to find a model of the subjective difficulty. In the first experiment, the most influential factors are evaluated. The result is then used in combination with the second experiment to find an optimized function for the perceived stroke difficulty.

For the first experiment, 25 participants (15male,21-31yrs,Ø25yrs) are shown 24 pool scenarios on a real pool table, see Appendix C. In every scenario two playable object balls are placed on the table. For 18 scenarios one of the three parameters (d_1, d_2, θ_c) and the AAD stroke difficulty are kept constant. For the other 6 scenarios, one of the two object balls is easier to pocket according to the AAD difficulty, but either more tricky to reach as the player has to lean more over the pool table or partially blocked by a third ball. Each participant has to judge which ball seems easier to be pocketed and is asked to specify freely one or more reasons for his decision which are categorized afterward. The most frequently stated reasons with their number of occurrence for the first 18 scenarios are shown in Tab. 4.1. The answers show that the variables d_1, d_2 and θ_c are the most influential factors. The effect of a cushion close to cue ball or object ball is indefinite as it helps some players at aiming whereas others feel disturbed. For the 6 remaining scenarios, in 70% the easier stroke according to the AAD method is selected even if it is

occurrence	reason
264×	A smaller cutting angle θ_c
75×	A smaller distance d_2
31×	A smaller distance d_1
29×	The cushion nearby helped at aiming
24×	The cushion nearby disturbed at aiming
12×	Intuition
9×	Anatomic reason
8×	A smaller distance $d_1 + d_2$
17×	Very specific reason

Table 4.1: Influence of different factors on the subjective difficulty

more tricky to reach. In 60% the easier stroke is selected even if it is partially blocked by another ball. Similar to the answers regarding the advantage of a cushion nearby, some people state that the ball helps at aiming, however other people feel distracted by it.

For the second experiment, 12 images are created showing pool scenarios of different difficulty according to the AAD method with one cue ball and one object ball, see App. C. For comparing every possible pair of images, six sets of six images are shown to 23 participants (13male,18-51yrs,Ø25yrs). The participants have to arrange every individual set of images according to their perceived difficulty. Based on these sequences, both a relative ranking R_n, $R, n \in \{1, 2, \ldots, 12\}$ and an absolute ranking ρ_n, $n \in 1, 2, \ldots, 12, \rho \in [1, 12]$ based on the mean ranking are obtained for every single scenario. In order to map the result of both experiments to a subjective difficulty, five different equations describing the subjective difficulty are considered:

$$\begin{aligned}
\kappa_1 &= a_1 d_1^{c_1(d_1)} + a_2 d_2^{c_2(d_2)} + a_3 \theta_c^{c_3(\theta_c)}, \\
\kappa_2 &= a_1 d_1^{c_1(d_1)} + a_2 d_2^{c_2(d_2)} + a_3 \theta_c^{c_3(\theta_c)} + \kappa_p, \\
\kappa_3 &= a_1 d_1^{c_1(d_1)} + a_2 d_2^{c_2(d_2)} + a_3 cos(\theta_c)^{c_3(\theta_c)}, \\
\kappa_4 &= a_1 d_1^{c_1(d_1)} + a_2 d_2^{c_2(d_2)} + a_3 cos(\theta_c)^{c_3(\theta_c)} + \kappa_p, \\
\kappa_5 &= \frac{cos(\theta_c)^{c_3(\theta_c)}}{d_1^{c_1(d_1)} d_2^{c_2(d_2)}},
\end{aligned}$$

with $\mathbf{a} = [a_1, a_2, a_3]$ as constant coefficients and $\mathbf{c} = [c_1, c_2, c_3]$ as polynomials of degree one. The first two measures are based upon additive linear models. The next two models include the term $cos(\theta_c)$ to account for the difficulty at high cutting angles. The last term is an extension of the existing measure (4.4). A two-staged optimization is used to find optimal values for $\mathbf{a}$ and $\mathbf{c}$. The cost function Γ_1

$$\Gamma_1 = \sum_{m=1}^{18} \delta_m(\mathbf{a}, \mathbf{c})^2 + \sum_{n=1}^{12} [\hat{R}_n(\mathbf{a}, \mathbf{c}) - R_n]^2, \qquad \delta_m = \begin{cases} 2 & \text{if } \hat{\Delta}_m(\mathbf{a}, \mathbf{c}) \neq \Delta_m, \\ 0 & \text{else,} \end{cases}$$

of the first stage consists of two terms related to the first and second experiment. Minimizing the first term corresponds to matching the predicted decision $\hat{\Delta}_m(\mathbf{a}, \mathbf{c})$ to the

decision Δ_m of the majority of participants in the first experiments. The factor δ_m acts as a penalty term if the predicted decision does not match the decision from the first experiment. By minimizing the second term, one matches the predicted rank $\hat{R}_n(\mathbf{a},\mathbf{c})$ to the actual rank R_n of each scenario in the second experiment depending on its difficulty. The cost function Γ_2 for the second stage

$$\Gamma_2 = \sum_{n=1}^{12} [\hat{\kappa}_n(\mathbf{a},\mathbf{c}) - \rho_{n,25}]^2,$$

is a measure how much the predicted absolute ranking $\hat{\kappa}_n(\mathbf{a},\mathbf{c})$ of each scenario in the second experiment differs from the 25% trimmed mean $\rho_{n,25}$. By minimizing the function Γ_2, a measure of the resulting stroke difficulty is obtained.

An intermediate calculation indicates that the parameters c_1, c_2 can be chosen as constant coefficients. Final results are displayed in Tab. 4.2, showing that the function κ_5 is the best approximation. Thus the subjective difficulty κ is determined as

measure	a_1	a_2	a_3	c_1	c_2	c_3	Γ_2
κ_1	4.6	7.1	1.2	0.10	0.10	$4.5-4.7\theta_c$	0.02
κ_2	1.3	6.5	2.0	1.0	0.21	$3.8-4.9\theta_c$	0.008
κ_3	-0.53	1.3	-4.8	-1.1	4.6	$0.24+2.2\theta_c$	0.06
κ_4	0.81	-1.4	-4.8	1.9	-0.75	$2.8+1.9\theta_c$	0.08
κ_5	-	-	-	0.33	0.38	$4.1-2.7\theta_c$	0.002

Table 4.2: Optimized parameters when approximating the subjective difficulty

$$\kappa = \frac{cos\theta_c^{\,4.1-2.7\theta_c}}{d_1^{0.33} d_2^{0.38}}.$$

Objective difficulty

Calculating the objective difficulty means finding the percentage $P_o \in [0;1]$ to pocket a ball. To get a good approximation over a wide variety of (θ_c, d_1, d_2)-triplets, a Monte-Carlo approach is used which evaluates random strokes during a game.

In total, four participants (3male,24-36yrs,Ø29yrs) take part in the third experiment. They are chosen according to their playing competence: Two players are intermediate amateurs who frequently play a few pool games per week, two are novice players with hardly any experience. Participants are grouped into dyads of equal playing skill. Their task is to play multiple games while sticking to direct strokes and hitting the cue ball centrally. Every measured stroke is assigned to one of eleven intervals depending on their subjective difficulty, ensuring a roughly similar number of strokes in every interval. To compare the different measures of subjective difficulty, they are normalized to the $[0;1]$ interval. For every interval, the success rate (objective difficulty) and mean subjective difficulty are calculated. Because the number of element per interval varies for every subjective difficulty measure, so does the objective difficulty vary from graph to graph.

player	evaluated strokes	correlation coefficient		
		κ_l	AAD	κ
one	216	0.46	0.61	0.89
two	182	0.65	0.72	0.86
three	164	0.54	0.58	0.88
four	152	0.76	0.68	0.92

Table 4.3: Correlation coefficient between subjective difficulty and objective difficulty when considering all strokes

Pearson's correlation coefficient between subjective difficulty κ and objective difficulty P_o for the three difficulty measures is shown in Tab. 4.3. While there is a linear relation between the subjective difficulty and objective difficulty for most strokes, for easy strokes the subjective difficulty becomes very large whereas the objective difficulty saturates at 1. Thus the correlation coefficient has been recalculated as shown in Tab 4.4, neglecting the interval with the easiest strokes. The linear relation when neglecting very easy strokes

player	evaluated strokes	correlation coefficient		
		κ_l	AAD	κ
one	169	0.79	0.85	0.94
two	152	0.87	0.77	0.91
three	134	0.64	0.52	0.88
four	121	0.89	0.82	0.91

Table 4.4: Correlation coefficients between subjective difficulty and objective difficulty when neglecting very easy strokes

has important consequences: It shows that humans act indeed rationally when selecting the subjective easiest stroke as it is the one with the highest chance to pocket a ball. On the other hand conclusions about which stroke a human selects are deduced from the objective difficulty. Fig. 4.4 displays results for one player, showing both the 10 respectively 11 intervals together with the calculated linear regression function for the mapping between subjective difficulty and objective difficulty. The three different relations correspond to the expert method, the AAD method and the human model developed in this section. The resulting function for the objective difficulty for player one thus becomes

$$P_o = \begin{cases} 0.36\kappa + 0.27 & \forall\, \kappa \in [0; 2.02], \\ 1 & \forall\, \kappa \in]2.02; \infty[. \end{cases}$$

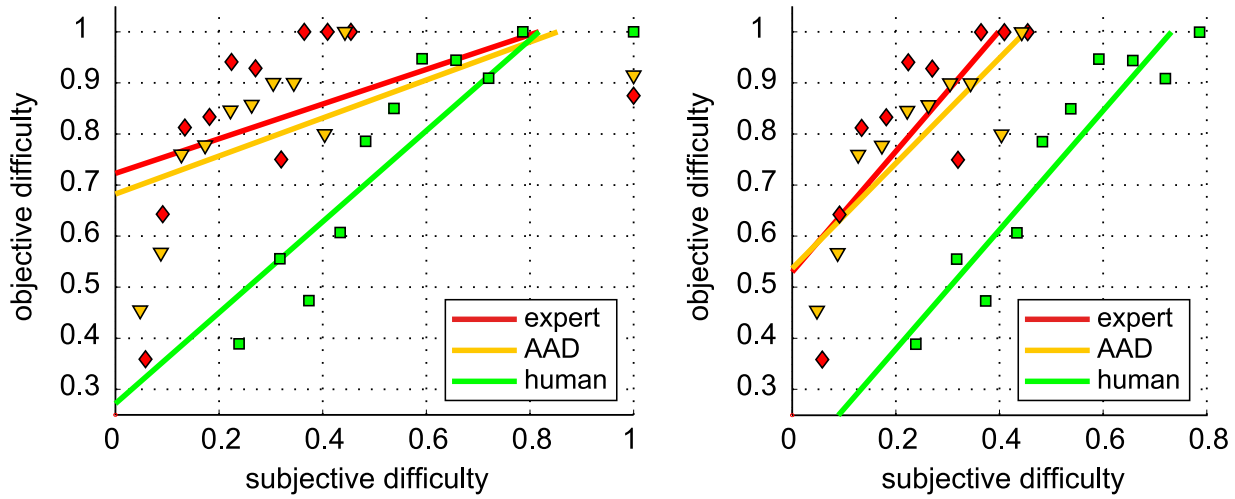


Figure 4.4: Normalized subjective difficulty vs. objective difficulty with corresponding regression lines. Left: All strokes considered. Right: Neglecting very easy strokes

Discount factor

The data from the third experiment is used to calculate an individual discount factor γ_h for every participant through a maximum likelihood estimation (MLE) maximizing the percentage of correct predictions Ψ for every player over the parameter γ_h as

$$\hat{\gamma}_h = \underset{\gamma_h}{\operatorname{argmax}} L(\gamma_h|\Psi).$$

Results of the MLE for two dyads for a planning depth of 1 are displayed in Fig. 4.5. When being asked, the intermediate player attests to think about the outcome of a stroke which corresponds well to the maximum for $\hat{\gamma}_h \approx 0.55$. The novice player on the other hand confirms focusing solely on the easiest stroke, a statement verified by the maximum for $\hat{\gamma}_h \approx 0$.

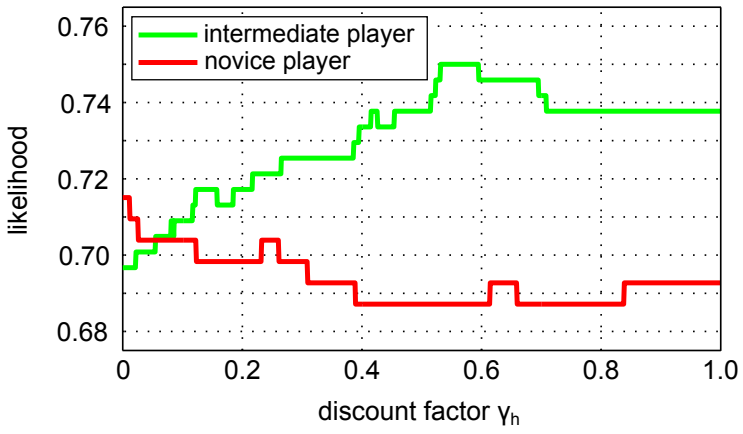


Figure 4.5: Maximum likelihood estimation of the human discount factor γ_h for a novice player and an intermediate player

4.7 Experimental evaluation

Both simulations and real-life experiments involving an anthropomorphic robot facing a human opponent on a real pool table evaluate the reasoning capabilities of the presented framework. Simulations allow for a fast evaluation based on a large number of strokes and provide holistic insights. The real world scenario on the other hand displays the applicability when facing a real human. For both human and robot the skill parameters are determined through multiple strokes recorded by the ceiling-mounted camera. They are and show that the human has got better pocketing skill, represented by smaller σ_p, σ_θ

parameter	robot	human	unit
σ_p	0.03	0.01	Ns
σ_θ	0.012	0.07	rad
γ	1	0.5	-

Table 4.5: Parameters for the experimental evaluation

values but worse planning capabilities as its γ value is smaller.

Simulations

To judge the planning and reasoning capabilities of different aspects of the framework independently, simulations evaluate the outcome of multiple played pool games to find significant improvements. The first simulation investigates the influence of a varied planning depth. As discussed in [13], the impact of a search depth of 2 is still an unsolved question. Differing from entirely deterministic games like chess or backgammon where the search depth is among the most influential factors, the stochastic nature of pool makes it difficult to answer this question. Depending on the skill of each player, deep search depths provide no additional insights or even has an adverse effect if the rewards are not well adjusted and the player focuses on the wrong objective. In [13] it is found out that both a search depth of two and a more densely sampled action space of depth 1 have a similar influence on the percentage to win. Simulations of 200 strokes for arbitrary pool states with a search depth of 0, 1 and 2 are evaluated to answer the question. As the first simulation scenario neither considers the human model nor any kinematic constraints, it is comparable to the experiment in [13]. In contrast, both human model and kinematic constraints are incorporated in the second scenario. Results are shown in Fig. 4.6. The colored circles are Venn diagrams, showing the number of similar/different object-ball/pocket decisions. As such for 17 strokes the planning results for a depth of 0 and 1 are similar for the first scenario, meaning that the same ball will be pocketed in the same pocket. Comparable to [13], most of the planned strokes - 105 for the first scenario, 77 for the second scenario - are similar regardless of the search depth. Moreover, a varied search depth changes the decision only in about 20% of all cases. Even if this provides no significant results about the outcome of a game, results are expected to be similar: Even if a search depth of 2 is beneficial, the biggest improvement comes from the available computation time and number of simulations to be evaluated. A planning depth of 1 or

2 plays only a minor role. A second simulation evaluates the effect of the human model.

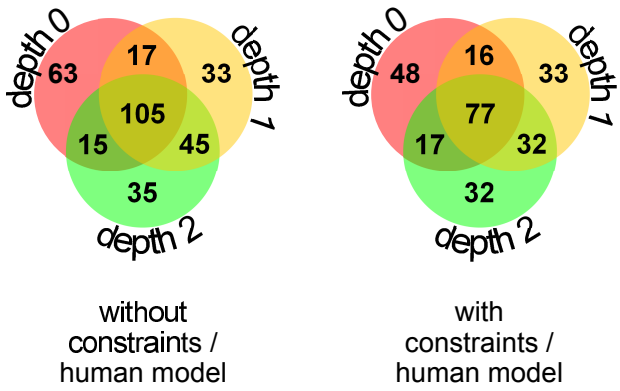


Figure 4.6: Influence of a varied planning depth for 200 random table states. Left: Ignoring kinematic constraints and the human model. Right: Considering both effects

The simulation consists of three different scenarios, depending on the way the robot's kinematic constraints as described in Sec. 4.5 are treated:

- without repositioning: Whenever the robot is unable to reach a ball, it is considered as a foul
- with repositioning: Whenever the robot is unable to reach a ball, the white ball is slightly moved such that the robot is able to execute a legal stroke
- without constraints: There are no kinematic constraints, i.e. the robot is able to reach any position on the table

Every scenario consists of two runs, one without considering the human model according to Sec. 4.6, another one where it is taken into account. In every run 1200 games are evaluated twice, both with robot and human as starting player. Results are checked for significance using a two-sample t-test and shown in Fig. 4.7 in addition to the number of won robot games. Two things are visible. Limiting the workspace of the robot by considering kinematic constraints has a major impact on the success rate of the robot. Without any constraints the robot has an approx. 30% chance to win the game, mainly limited by the skill of the robot caused by hardware inaccuracies. When considering the kinematic constraints this rate drops to slightly more than 10%, yet a more advanced hardware setup may overcome this problem. The more important aspect is that considering the human model significantly increases the success rate as the robot wins about 70 games more compared to the version where it does not consider thee human model. When planning ahead it is important to provide the planner with a precise model of the kinematic constraints so that the robot can continue playing after pocketing a ball. A third simulation is conducted and similar to the previous simulation 1200 games are evaluated twice with human and robot as starting player. The kinematic constraints of the robot are treated in two different ways:

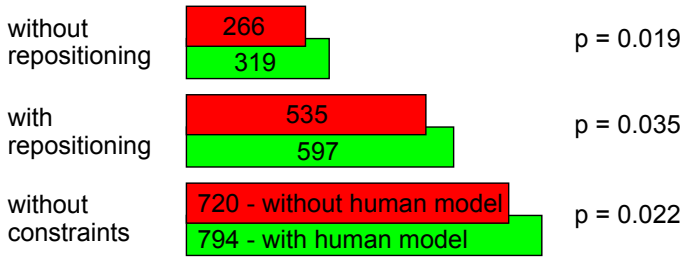


Figure 4.7: Influence of the human model and the kinematic constraints at pool. Left: Number of robot wins for different scenarios after 2400 simulated games. Right: Significance level of every comparison using a two-sample t-test

- fake repositioning: When planning ahead, no kinematic constraints are considered. During stroke execution, it is considered as a foul whenever the robot is unable to reach a ball.

- without repositioning: When planning ahead, kinematic constraints are considered. During stroke execution, it is considered as a foul whenever the robot is unable to reach a ball.

Whereas for the “without repositioning” case the model of the kinematic constraints resembles the real situation, the “fake repositioning” case ignores the kinematic constraints for the model, yet they account in reality. The latter case corresponds to a very bad modeling of the true kinematic constraints. Simulations are performed for a robot planning depth of 0 and 1 to show how the influence of an increased planning depth depends on the fidelity of the underlying model. Results are shown in Fig. 4.8 together with the significance level of a two-sample t-test. The results for the “fake positioning” case do not differ significantly as the planner has a wrong model of the kinematic constraints. This nullifies the effect of planning ahead to a large extent and results in roughly similar results for different planning depths. The second case, which is identical to the first case in Fig. 4.6, is rerun for a planning depth of 0 and evaluated. As the robot has a precise model of the kinematic constraints during the game, a planning depth of 1 significantly improves the robot’s position play.

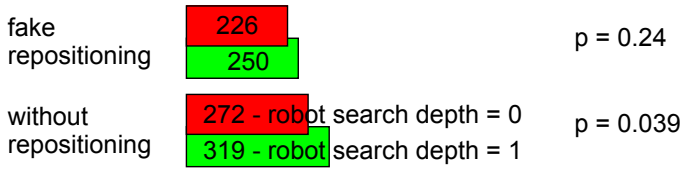


Figure 4.8: Influence of the kinematic constraints model at pool. Left: Number of robot wins for different planning depths after 2400 simulated games. Right: Significance level of every comparison using a two-sample t-test

Real life duel with an anthropomorphic robot

The hardware setup for the real-life experiment consists of an anthropomorphic robot with a pair of 7DOF arms mounted on an omnidirectional platform, see [40]. Special gimbal-based EEs are designed to hold the cue properly. A ceiling-mounted camera (Basler acA1300-30gm) with a resolution of $1280 \times 960px$ mounted approx. $2.50m$ above the table detects the position of all balls and the cue relative to the table and is able to distinguish between white, black, striped and solid balls. In order to place the cue properly behind the cue ball, combined data of the camera, the robot's laser range finders and arm pose data is used. The robot is able to move autonomously around the table by detecting the legs of the pool table and execute a stroke independently. Fig. 4.9 gives an image of the scenario. A detailed scheme of the combined hardware/software

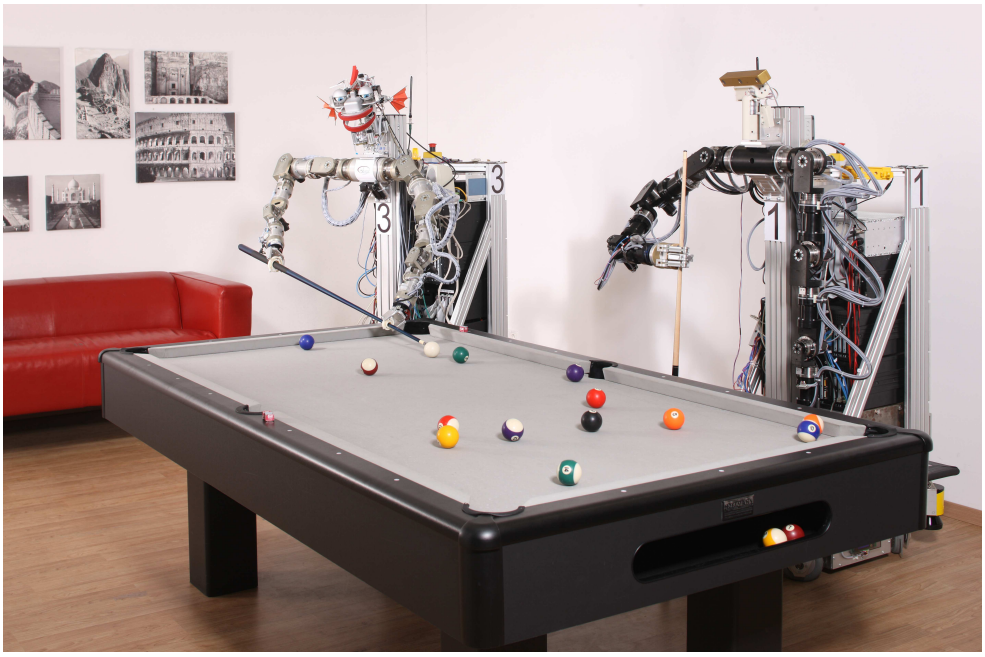


Figure 4.9: Pool scenario overview. Photo by Kurt Fuchs

setup is displayed in Fig. 4.10. Two computers with a $2.66GHz$ Intel Pentium i7 920 CPU and $12GB$ RAM on the robot control both arms and the platform at an update rate of $1000Hz$. A Real-time database [11, 102] acts as a shared memory and allows data exchange while sufficing hard real time constraints. Both computers are connected via Ethernet. The arm controller is programmed in Matlab/Simulink and compiled using MathWork's Realtime Workshop. The platform controller is programmed in C++. Motion planning is performed on the central PC equipped with a AMD Phenom II X6 1075T $3GHz$ CPU and $8GB$ RAM under ROS [249], processing sensor data from the camera, the robots and the boundary constraints from the motion reasoning scheme to calculate a feasible motion for executing a stroke. It also features a state machine to move the robot around the table and place the cue behind the cue ball before the stroke. The update rate for arm and platform data is $10Hz$, for the camera data it varies between $40Hz$ when using a $640 \times 480px$ resolution and $10Hz$ for the full resolution of $1280 \times 960px$.

The central PC exchanges data with the stroke planning PC about the position of all balls on the table and the boundary constraints for motion planning as described in Sec. 4.5 via shell scripts. Once the stroke planning PC has information about the ball position, it calculates boundary constraints for an optimal motion as described in Sec 4.4 in Matlab and sends the results to the central PC. A C++\Qt based pool simulator is developed and calibrated as described in App. C and [222, 230], predicting the outcome of a stroke and forming the foundation to calculate an optimal stroke of the robot. To speed up calculations, computations are parallelized on a cluster of 40 computers with a AMD Phenom II X6 1075T 3GHz CPU and 8GB RAM. The planning capabilities of the robot are evalu-

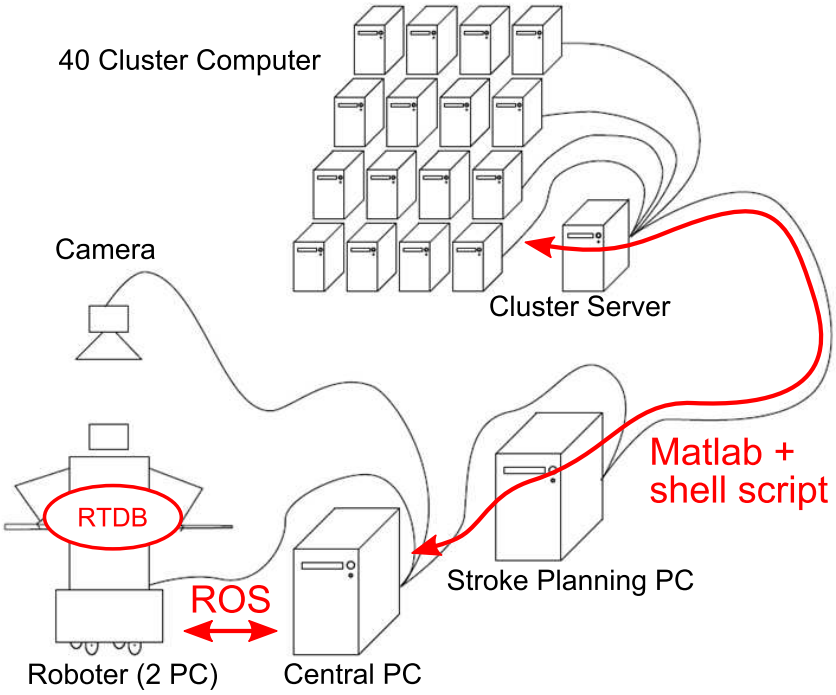


Figure 4.10: Pool robot system overview]

ated by comparing two scenarios with the same human opponent. In the first scenario a planning depth of 0 and no human model is used. A manually set stroke intensity p of 0.17Ns results in an initial cue ball speed of $1\frac{m}{s}$. In total 7 games are played. Although the number of played games is too low to make any statement about the effectiveness of the planner, temporal limitations make an appropriate number of rollout impossible. The authors of [13] for example simulated more than 600 games before achieving significant results, which would require too much time with a real pool robot. Instead the number of successful strokes is evaluated. Implicitly it is assumed that a large number of successful strokes leading to more pocketed balls also increases the probability to win the game. The 7 games result in 89 strokes of the robot, out of which 25 are successful. In the scenario the planning depth is set to 2 and the human model is considered. Here 6 games are played, resulting in 67 strokes of the robot. Out of these 67 strokes 34 are successful. A two-sample t-test shows a significant difference between the number of successful strokes, indicating that the inclusion of the human model, a precise model of the kinematic constraints and a deeper sampling depth lead to an improved game play of the

robot. When comparing results from both simulations and the real experiment, hardware

Figure 4.11: Pool robot evaluation. Left: Number of successful strokes for different search depths using an anthropomorphic robot. Right: Significance level of every comparison using a two-sample t-test

inaccuracies and kinematic constraints constitute the biggest impairment for the robot. Although the robot weights about $300kg$, the platform shaking during the stroke causes cue tip deviations. Even if this is partially compensated by using low accelerations during the stroke, the resulting offset of the cue is still in the range of $0,5cm$, making a central hit of the cue ball difficult. Although the kinematic constraints of the robot look sever at first glance, they are overcome to a certain extent by small hardware inaccuracies and a precise model of the environment through a well-matched pool simulator.

It has to be highlighted that the tight coupling between a player's skill in terms of σ_θ, σ_p and its planning capabilities represented by the maximum planning depth γ is a quite unique problem at pool as both aspects have a large effect on motion reasoning about the next optimal stroke. A counterexample would be chess where motion reasoning is solely influenced by the planning capabilities of each player, the player's skill is not important in this context. This makes it also difficult to reason the other way round as one cannot tell whether a player has superior planning or skill capabilities by just looking at the outcome of the game. As shown before it is necessary to differentiate between subjective difficulty and objective difficulty for motion reasoning in a complex scenario. Whereas the first characterizes the user's personal preference, the latter is an objective measure about the outcome of each action. Although it was not the case for the given scenario, there is not necessarily a clear relation between the two measures. In this sense classical motion reasoning schemes based upon expert knowledge solely model the subjective difficulty perceived by an expert. For nondeterministic tasks where the user has an influence on the outcome of a stochastic process, it is additionally required to model the objective difficulty, either independently or as a function of the subjective difficulty. General as the presented approach is, it can be transferred to other nondeterministic tasks containing only a single agent or multiple agents interacting both in a cooperative or competitive manner.

4.8 Summary

Motion reasoning in the field of motion adaptation is a necessary tool if a task is too big to be completed by a single adapted motion. Through separation into smaller motions it provides an efficient way to find an optimal motion while reducing computational

complexity. Whereas motion reasoning in its general context covers a variety of different fields including sensing, task, data and knowledge description or suitable abstraction and complexity reduction schemes, this thesis focuses on modeling the environment in a holistic way while considering the individual preferences of a human opponent. The used data representation as a MDP is a generic way and accounts for a multiplicity of different tasks. It becomes clear that motion reasoning must not be seen as an independent topic completely decoupled from motion adaptation, rather it has to be aware of the intrinsic capabilities and limitation of motion adaptation approach through a proper model of the task.

4.9 Bibliographical notes

The results in this chapter and the related appendix are either partly or in their entity based on the author's work in [188, 222, 230]. The general approach of using an MDP to encode the game of pool is not new but has been also adopted in [13]. The main difference between their work and ours is the used heuristics for making the problem mathematically tractable, the type of tree search (breadth-first vs. Monte-Carlo) and the used measure to calculate the stroke difficulty. Whereas the AAD measure is only mentioned in other articles but not used, the expert method is used in literature to a great extent [76, 177, 192, 276]. The underlying pool simulator as illustrated in App. C is developed to a large extent based on the results presented in [112].

5

Conclusions

The increasing complexity of robotic systems and tight interaction between humans and robots requires a rethinking of how to program robot motions. Differing from a conventional approach where every motion primitive is specified manually, PbD accelerates the teaching process and allows complex robotic movements to be executed. This thesis investigates the adaptation capabilities of trajectory-encoded movement imitation schemes, allowing real-time adaptation in changed environments. Special focus is drawn on a holistic approach that ensures consistency of the motion reasoning, planning and control domain by pursuing similar objectives across the three domains.

5.1 Contributions

Focusing on motion planning in cluttered environments, Sec. 2 investigates methods to generate feasible motions for complex manipulators up to a 50-DOF robot. To avoid the curse of dimensionality during planning, the programming by demonstration paradigm is adopted. Instead of planning every motion from scratch, a previously taught motion is modified to suffice the new constraints. Based upon Laplacian Mesh Editing in computer graphics, a new framework named LTE is presented that allows to quickly retarget discretized motions. Through least squares it provides a weighted solution between the shape of the original path and additional constraints, providing an intuitive understanding and allowing the . The section also shows how different constraints for collision avoidance, cooperative manipulation and nonlinear deformation effects can be included in the least squares solution and thus handled efficiently. Approximations schemes for the true solution are presented, speeding calculations up to three magnitudes and making the approach applicable for large trajectories and highly redundant manipulators.

As conventional gradient-based collision avoidance schemes fail in highly cluttered environments, LTE is combined with an incremental, sampling-based approach (RRT*) for better performance. Three different experiments in real-life validate the effectiveness of the proposed approach, allowing fast motion retargeting within milliseconds and the applicability to a 50-DOF robot.

In Sec. 3 focus shifts from motion planning to robot control after a feasible reference trajectory is found. The goal becomes to control a robot in its entity while considering task- and LTE-specific constraints for an integrated motion planning and control framework. To account for the predictive nature of LTE, a LQR-based controller is combined with a prioritized IK approach for full-body control. Two different extensions allow the combination with LTE. The first approach extends the state vector of a LTI system and considers the local properties of a previously calculated reference trajectory, resulting in a serial connection of LTE and the LQR controller. The second approach combines LTE and LQR in a feedback control structure, iteratively calculating an optimal input at each time step and updating the desired trajectory accordingly. This allows the combination of motion control and planning, ensuring convergence towards the reference trajectory while being able to react to sudden disturbances, include additional constraints and be consistent with the LTE framework. Asymptotic stability for both controllers is proven. If the manipulator has a non-negligible weight and inertia, its dynamic properties must be considered as well. By deriving similarities between LTE and a spline deformation approach, upper bounds on the admissible trajectory deformation subject to dynamic constraints are calculated. During robot control, the discrete paths used by LTE are only one option to encode a motion. As the motion reproduction capabilities of a manipulator depend highly on the choice of motion representation, three different motion encoding schemes are presented, focusing on different motion aspects. As such they investigate the motion classification and reproduction qualities subject to rigid transformations, a probabilistic and entirely distance-based task space encoding and a continuous interpolation scheme based on vector fields. All three methods provide very good results in their field, verified through simulations and experiments with a HRP-4 humanoid.

Sec. 4 considers motion reasoning, investigating the relation between actions and their outcome - not only in a static environment but also when interacting with other persons in a competitive scenario. By encoding a task as a Markov decision process the temporal dependence between actions of the robot and resulting states of the environment is specified. A model-based approach allows to generate an optimized action from the simulated outcome of each action which can be processed by the underlying motion adaptation scheme. To model all actions properly in the presence of other interacting humans, a precise model of the human behavior is necessary. Using the game pool as a representative scenario for turn-based interaction between human and robot, the influence of the human model is investigated in detail through psychological experiments and model fitting. Results based on simulations and a anthropomorphic robot facing a human opponent at a real pool table show significant improvements when using a proper model.

5.2 Outlook

Maintaining local trajectory properties during motion adaptations creates new challenges during motion adaptation that must be tackled at different abstraction levels. This section covers some emerging issues for future expansion which are treated either on a basic level or not at all in this work.

Motion representation

The used encoding determines both the intrinsic capabilities and limitations of the motion adaptation approach. Encoding schemes differ in the way they encode data (deterministic vs. probabilistic), the type of encoded data (single limb vs. whole body) and the type of constraints one can impose (temporal vs. spatial, collision avoidance, waypoints, velocity/acceleration limits, etc). Due to the novelty of PbD, there is not yet one uniform approach at hand capable of handling all motion adaptation related requirements. LTE also suffers from a few drawbacks (e.g. being limited to discrete trajectories) and is thus not the definite solution. At the moment it is unknown whether an existing approach can be extended to include all relevant constraints or a novel encoding scheme has better generalization capabilities and fewer limitations.

Rotations

The least squares solution of LTE is used in this thesis to deform the position of an EE or other robot links. Rotations are only implicitly treated through the underlying kinematic chain. Although there are methods for a spline-based rotation interpolation available [239, 247] they account only for specific cases and not a generic least squares solution. It is thus still an open question how to include rotations in task space in the existing LTE formalism.

Sensor-based constraint specification

Although LTE can cope with a variety of constraints, this thesis focuses on how such constraints can be incorporated to adapt a given path when necessary. It is still unknown how these constraints can be automatically extracted from sensor data to allow safe task execution. Arising question include but are not limited to: Which sensors to use? Which sensor fusion algorithm will provide most information about the environment at minimal computational cost? Which sensor information can be processed efficiently by LTE? What is the minimal number of constraints that must be specified for fulfilling the given task?

Real-time reasoning for human-robot-interaction

The presented motion reasoning scheme relies on a Markov decision process, is applicable to a wide class of problems and provides a near-optimal solution through numeric

optimization. For turn-based scenarios the high computational complexity of the backward induction approach is not a critical issue. However, if the robot has to reason about an optimal motion when interacting directly with a human, calculation times in the range of seconds or even minutes are not an option and require a different approach to reason about which action to perform in which state.

Cognition for automatic gain knowledge

LTE can be seen as a small step towards a humanoid robot capable of learning and adapting new motions autonomously. In this case focus shifts from “how” to “why” a movement should be adapted. Whereas in this thesis both the prototypic reference motion and constraints are specified manually for both scenarios, a cognitive robot might be able to acquire such knowledge simply by observing others’ movements and drawing its own conclusions. Hardly any research has been done so far in the field of PbD and possible investigations spans across the domains of engineering, neuroscience and psychology.

A

Appendix to chapter 2

Path similarity measure

Fig. A.1 shows all ten reference paths of the psychological experiment to determine the human perception of LTE. For each set of paths the black path denotes the undeformed path whereas green, orange and red paths undergo different amounts of deformation. The upper row lists the five synthetic paths, the lower row the real paths recorded with a Kinect. The questionnaire handed out to each participant of the experiment is shown

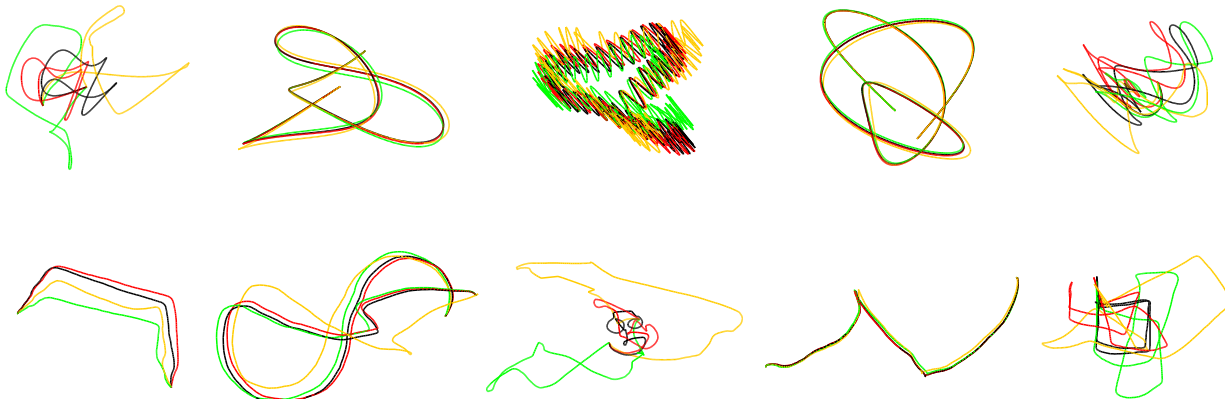
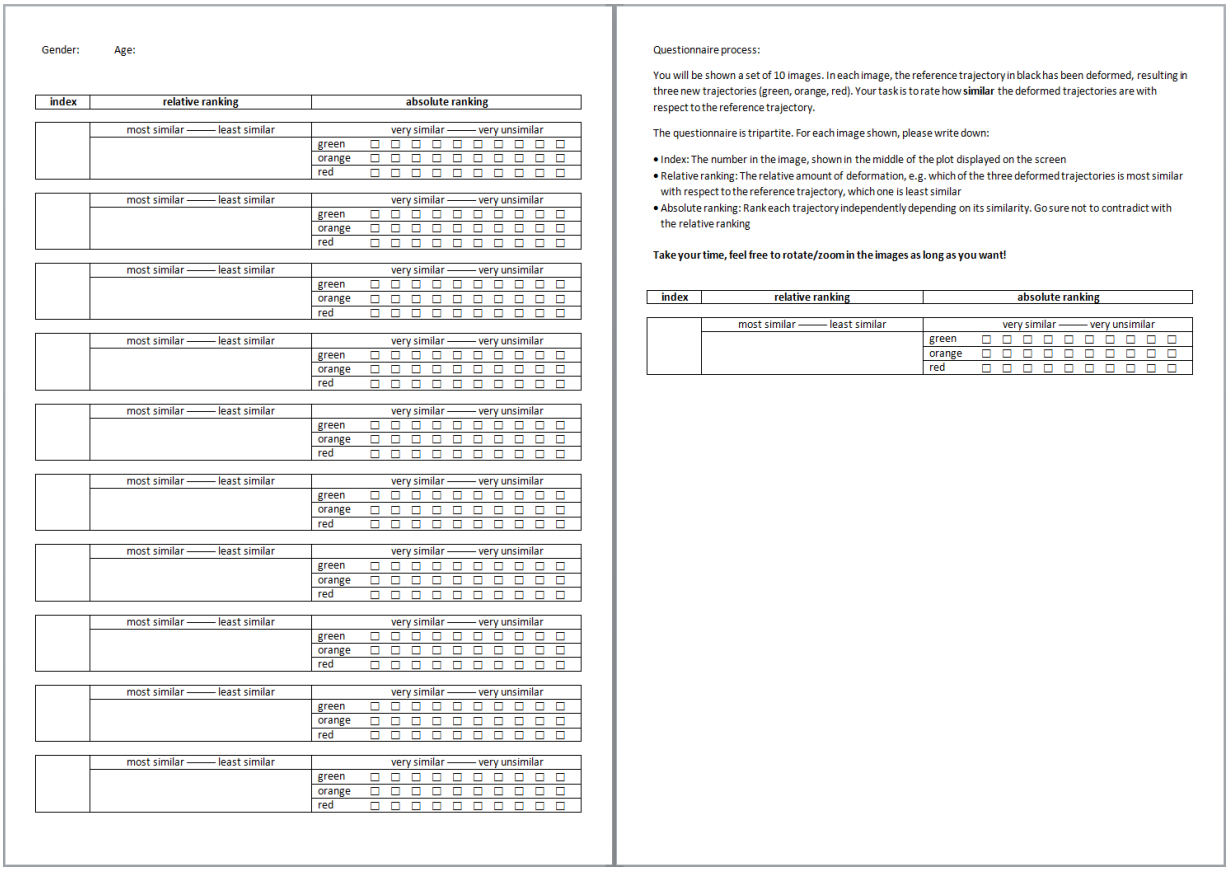


Figure A.1: Reference and deformed paths for path similarity evaluation

in Fig. A.2



Gender: Age:

index	relative ranking	absolute ranking
	most similar —— least similar	very similar —— very unsimilar
		green □ □ □ □ □ □ □ □ □ □
		orange □ □ □ □ □ □ □ □ □ □
		red □ □ □ □ □ □ □ □ □ □

(repeated 10 times)

Questionnaire process:

You will be shown a set of 10 images. In each image, the reference trajectory in black has been deformed, resulting in three new trajectories (green, orange, red). Your task is to rate how **similar** the deformed trajectories are with respect to the reference trajectory.

The questionnaire is tripartite. For each image shown, please write down:

- Index: The number in the image, shown in the middle of the plot displayed on the screen
- Relative ranking: The relative amount of deformation, e.g. which of the three deformed trajectories is most similar with respect to the reference trajectory, which one is least similar
- Absolute ranking: Rank each trajectory independently depending on its similarity. Go sure not to contradict with the relative ranking

Take your time, feel free to rotate/zoom in the images as long as you want!

index	relative ranking	absolute ranking
	most similar —— least similar	very similar —— very unsimilar
		green □ □ □ □ □ □ □ □ □ □
		orange □ □ □ □ □ □ □ □ □ □
		red □ □ □ □ □ □ □ □ □ □

Figure A.2: Path similarity measure questionnaire

Volleyball scenario

For the volleyball scenario in Sec. 2.10.1 a precomputed optimal trajectory is deformed online to bat the puck in such a way it follows the desired trajectory. As the derivation of the optimal batting motion and the resulting trajectory deformation is not that straightforward as it seems, it is recapitulated below. The goal of the impact modeling is to find the full state vector of the endeffector for the desired puck motion, described by the hit position (x_h, y_h) along the table, the fixed endeffector angle Φ_h and its translational velocity (v_{xh}, v_{yh}). After those information are calculated, they can be incorporated in LTE through suitable positional/velocity constraints. For a tilted airhockey table the puck moves along a parabolic trajectory described by

$$\begin{aligned} x_h &= x_0 + v_{x0} t_h, \\ y_h &= y_0 + v_{y0} t_h - \frac{1}{2} g t_h^2. \end{aligned} \tag{A.1}$$

The preferred hit point (x_h, y_h) lies on a circle with constant radius r_c around the robot's base (x_c, y_c). The hit point at time t_h is then found by solving (A.1) subject to the constraint

$$(x_h - x_c)^2 + (y_h - y_c)^2 - r_c^2 = 0.$$

The velocity of the puck immediately before the hit is denoted by (v_x, v_y) and the rotational speed by ω.

To find the remaining unknowns, one has to specify the velocities (v'_x, v'_y) and ω' immediately after the hit. Similar to (A.1) the puck moves on a parabolic trajectory after the hit. By defining two waypoints (x_1, y_1) and (x_2, y_2) along the parabola the puck has to pass, one can solve the resulting system of equations

$$\begin{aligned}
x_1 &= x_h + v'_x t_1, \\
y_1 &= y_h + v'_y t_1 - \frac{1}{2} g t_1^2, \\
x_2 &= x_h + v'_x t_2, \\
y_2 &= y_h + v'_y t_2 - \frac{1}{2} g t_2^2,
\end{aligned}$$

for the four unknowns t_1, t_2, v'_x, v'_y. The variables t_1 and t_2 denote the time when passing the two waypoints. When knowing the state of the puck immediately before and after the hit, one can derive the required end-effector angle and velocity during the hit. To do so, the puck velocities are first decomposed into relative normal components $\mathbf{v}_\perp, \mathbf{v}'_\perp$ and relative tangential components $\mathbf{v}_\parallel, \mathbf{v}'_\parallel$ before and after the hit as

$$\begin{aligned}
\mathbf{v}_\perp &= \mathbf{P}_\perp \left(\begin{bmatrix} v_x \\ v_y \end{bmatrix} - \begin{bmatrix} v_{xh} \\ v_{yh} \end{bmatrix} \right), \quad \mathbf{v}_\parallel = \mathbf{P}_\parallel \left(\begin{bmatrix} v_x \\ v_y \end{bmatrix} - \begin{bmatrix} v_{xh} \\ v_{yh} \end{bmatrix} \right), \\
\mathbf{v}'_\perp &= \mathbf{P}_\perp \left(\begin{bmatrix} v'_x \\ v'_y \end{bmatrix} - \begin{bmatrix} v_{xh} \\ v_{yh} \end{bmatrix} \right), \quad \mathbf{v}'_\parallel = \mathbf{P}_\parallel \left(\begin{bmatrix} v'_x \\ v'_y \end{bmatrix} - \begin{bmatrix} v_{xh} \\ v_{yh} \end{bmatrix} \right),
\end{aligned}$$

with the projection matrices $\mathbf{P}_\perp, \mathbf{P}_\parallel$ depending on the endeffector angle Φ as

$$\begin{aligned}
\mathbf{P}_\parallel &= [\cos(\Phi_h) \sin(\Phi_h)]^T [\cos(\Phi_h) \sin(\Phi_h)], \\
\mathbf{P}_\perp &= \mathbf{I} - \mathbf{P}_\parallel.
\end{aligned}$$

Fig. A.3 gives a detailed illustration of the impact model. The hit between puck and endeffector is modeled as a partially elastic collision with restitution coefficient ϵ along the normal direction, resulting in the change of normal impulse $\Delta\mathbf{p}_\perp$. In tangential direction it is assumed that the change of impulse $\Delta\mathbf{p}_\parallel$ during the hit just reduces the relative tangential velocity $\mathbf{v}_{\parallel r}$ between puck and endeffector to zero. The tangential impulse also leads to an angular momentum $\mathbf{L}_\parallel$. Mathematically it is described by

$$\begin{aligned}
\Delta\mathbf{p}_\perp &= -(1+\epsilon)\mathbf{p}_\perp, \\
\Delta\mathbf{p}_\parallel &= -m\mathbf{v}_{\parallel r} = -m(\mathbf{v}_\parallel + \boldsymbol{\omega} \times \mathbf{r}), \\
\Delta\mathbf{L}_\parallel &= \mathbf{r} \times \Delta\mathbf{p}_\parallel,
\end{aligned}$$

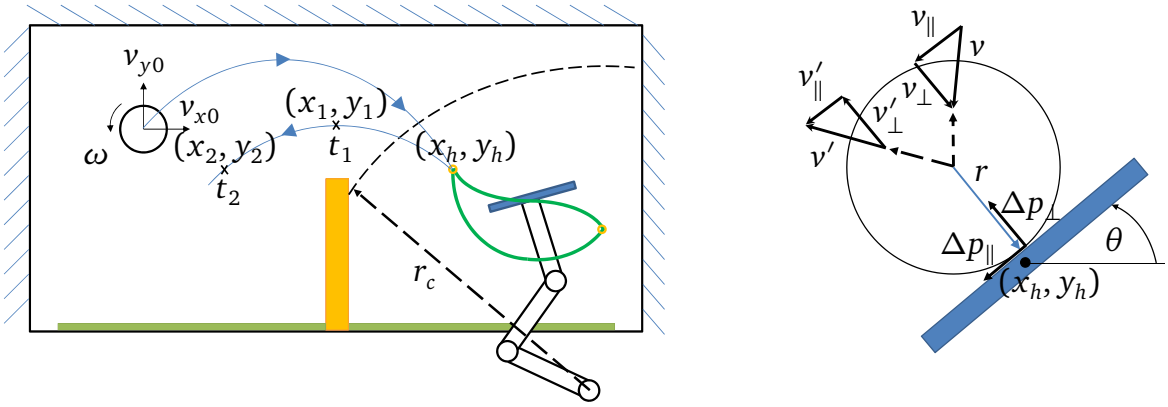


Figure A.3: Schematic view of the volleyball scenario. Left: Overview of the scenario. Right: Detailed impact model

with m as the mass of the puck, $\mathbf{r}$ as the vector from the center of the puck to the contact point, $\boldsymbol{\omega}$ as the vectorized version of ω and the operator $\times$ as the pseudo cross product in 2D. The resulting velocities before and after the hit can then be related to each other as

$$\begin{aligned}
\mathbf{v}'_\perp &= \frac{1}{m}(\mathbf{p}_\perp + \Delta\mathbf{p}_\perp) = -\epsilon\mathbf{v}_\perp, \\
\mathbf{v}'_\parallel &= \frac{1}{m}(\mathbf{p}_\parallel + \Delta\mathbf{p}_\parallel) = -\boldsymbol{\omega} \times \mathbf{r}, \\
\omega' &= \omega + \frac{1}{\Theta}\mathbf{r} \times \Delta\mathbf{p}_\parallel,
\end{aligned} \tag{A.2}$$

with Θ as the moment of inertia and $\mathbf{p}_\perp = m\mathbf{v}_\perp$, $\mathbf{p}_\parallel = m\mathbf{v}_\parallel$ as the normal/tangential impulse of the puck before the hit. Due to the nonlinearity of the projection matrices $\mathbf{P}_\parallel$ and $\mathbf{P}_\perp$, the resulting system of equations (A.2) has to be solved numerically for the variables v_{xh}, v_{yh}, Φ_h.

B

Appendix to chapter 3

Whole-body control using LQR/LTE

The cost function S_{k-1} can be calculated from S_k and the optimal input $\mathbf{u}_{k-1}$ in the general form

$$\begin{aligned} S_{k-1} = \mathbf{v}_{1,k-1}^T \Delta\hat{\mathbf{p}}_{k-1} + \mathbf{w}_{1,k-1}^T \Delta\check{\mathbf{p}}_{k-1} + \\ \Delta\hat{\mathbf{p}}_{k-1}^T \mathbf{v}_{2,k-1} + \Delta\check{\mathbf{p}}_{k-1}^T \mathbf{w}_{2,k-1} + \\ \Delta\hat{\mathbf{p}}_{k-1}^T \mathbf{X}_{k-1} \Delta\hat{\mathbf{p}}_{k-1} + \\ \Delta\hat{\mathbf{p}}_{k-1}^T \mathbf{Y}_{1,k-1} \Delta\check{\mathbf{p}}_{k-1} + \\ \Delta\check{\mathbf{p}}_{k-1}^T \mathbf{Y}_{2,k-1} \Delta\hat{\mathbf{p}}_{k-1} + \\ \Delta\check{\mathbf{p}}_{k-1}^T \mathbf{Z}_{k-1} \Delta\check{\mathbf{p}}_{k-1} + c_{k-1}. \end{aligned}$$

With the simplification of

$$\begin{aligned} \tilde{\mathbf{R}}_k &= \mathbf{R} + \mathbf{B}^T \mathbf{X}_k \mathbf{B}, \\ \tilde{\mathbf{M}}_k &= \mathbf{M} + \mathbf{B}^T \mathbf{Z}_k \mathbf{B}, \\ \tilde{\mathbf{Y}}_{1,k} &= \mathbf{B}^T \mathbf{Y}_{1,k} \mathbf{B}, \\ \tilde{\mathbf{Y}}_{2,k} &= \mathbf{B}^T \mathbf{Y}_{2,k} \mathbf{B}, \\ \tilde{\mathbf{C}}_1 &= (\mathbf{C}_1 - \mathbf{I}), \\ \tilde{\mathbf{C}}_2 &= (\mathbf{C}_2 - \mathbf{I}). \end{aligned}$$

and $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_4, \mathbf{c}_5$ as given in (3.18), the matrices $\mathbf{X}_{k-1}, \mathbf{Y}_{1,k-1}, \mathbf{Y}_{2,k-1}, \mathbf{Z}_{k-1}$, vectors $\mathbf{v}_{1,k-1}, \mathbf{w}_{1,k-1}, \mathbf{v}_{2,k-1}, \mathbf{w}_{2,k-1}$ and scalar c_{k-1} are calculated as

$$\begin{aligned}\mathbf{X}_{k-1} = {} & \mathbf{C}_3^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_3 + \\ & \mathbf{C}_3^T(\mathbf{B}^T\mathbf{X}_k\mathbf{A} + \mathbf{B}^T\mathbf{Y}_{2,k}\mathbf{A}) + \\ & (\mathbf{A}^T\mathbf{X}_k\mathbf{B} + \mathbf{A}^T\mathbf{Y}_{1,k}\mathbf{B})\mathbf{C}_3 + \\ & \mathbf{Q} + \mathbf{A}^T\mathbf{X}_k\mathbf{A},\end{aligned}$$

$$\begin{aligned}\mathbf{Y}_{1,k-1} = {} & \mathbf{C}_3^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_4 + \\ & \mathbf{C}_3^T(\mathbf{B}^T\mathbf{Y}_{1,k}\mathbf{A} + \mathbf{B}^T\mathbf{Z}_k\mathbf{A}) + \\ & (\mathbf{A}^T\mathbf{X}_k\mathbf{B} + \mathbf{A}^T\mathbf{Y}_{1,k}\mathbf{B})\mathbf{C}_4 + \\ & \mathbf{A}^T\mathbf{Y}_{1,k}\mathbf{A},\end{aligned}$$

$$\begin{aligned}\mathbf{Y}_{2,k-1} = {} & \mathbf{C}_4^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_3 + \\ & \mathbf{C}_4^T(\mathbf{B}^T\mathbf{X}_k\mathbf{A} + \mathbf{B}^T\mathbf{Y}_{2,k}\mathbf{A}) + \\ & (\mathbf{A}^T\mathbf{Y}_{2,k}\mathbf{B} + \mathbf{A}^T\mathbf{Z}_k\mathbf{B})\mathbf{C}_3 \\ & \mathbf{A}^T\mathbf{Y}_{2,k}\mathbf{A},\end{aligned}$$

$$\begin{aligned}\mathbf{Z}_{k-1} = {} & \mathbf{C}_4^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_4 + \\ & \mathbf{C}_4^T(\mathbf{B}^T\mathbf{Y}_{1,k}\mathbf{A} + \mathbf{B}^T\mathbf{Z}_k\mathbf{A}) + \\ & (\mathbf{A}^T\mathbf{Y}_{2,k}\mathbf{B} + \mathbf{A}^T\mathbf{Z}_k\mathbf{B})\mathbf{C}_4 + \\ & \mathbf{L} + \mathbf{A}^T\mathbf{Z}_k\mathbf{A},\end{aligned}$$

$$\begin{aligned}\mathbf{v}_{1,k-1}^T = {} & \hat{\mathbf{u}}_{k-1}^T\tilde{\mathbf{C}}_1^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k})\mathbf{C}_3 + \hat{\mathbf{u}}_{k-1}^T\mathbf{C}_1^T(\tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_3 + \\ & \check{\mathbf{u}}_{k-1}^T\mathbf{C}_2^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k})\mathbf{C}_3 + \check{\mathbf{u}}_{k-1}^T\tilde{\mathbf{C}}_2^T(\tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_3 + \\ & \mathbf{c}_5^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k})\mathbf{C}_3 + \mathbf{c}_5^T(\tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_3 + \\ & (\tilde{\mathbf{C}}_1\hat{\mathbf{u}}_{k-1} + \mathbf{C}_2\hat{\mathbf{u}}_{k-1} + \mathbf{c}_5)^T\mathbf{B}^T\mathbf{X}_k\mathbf{A} + \\ & (\mathbf{C}_1\check{\mathbf{u}}_{k-1} + \tilde{\mathbf{C}}_2\check{\mathbf{u}}_{k-1} + \mathbf{c}_5)^T\mathbf{B}^T\mathbf{Y}_{2,k}\mathbf{A} \\ & (\mathbf{v}_1 + \mathbf{w}_1)^T\mathbf{B}\mathbf{C}_3 + \mathbf{v}_1^T\mathbf{A},\end{aligned}$$

$$\begin{aligned}\mathbf{w}_{1,k-1}^T = {} & \hat{\mathbf{u}}_{k-1}^T\tilde{\mathbf{C}}_1^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k})\mathbf{C}_4 + \hat{\mathbf{u}}_{k-1}^T\mathbf{C}_1^T(\tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_4 + \\ & \check{\mathbf{u}}_{k-1}^T\mathbf{C}_2^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k})\mathbf{C}_4 + \check{\mathbf{u}}_{k-1}^T\tilde{\mathbf{C}}_2^T(\tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_4 + \\ & \mathbf{c}_5^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k})\mathbf{C}_4 + \mathbf{c}_5^T(\tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_4 + \\ & (\tilde{\mathbf{C}}_1\hat{\mathbf{u}}_{k-1} + \mathbf{C}_2\hat{\mathbf{u}}_{k-1} + \mathbf{c}_5)^T\mathbf{B}^T\mathbf{Y}_{1,k}\mathbf{A} + \\ & (\mathbf{C}_1\check{\mathbf{u}}_{k-1} + \tilde{\mathbf{C}}_2\check{\mathbf{u}}_{k-1} + \mathbf{c}_5)^T\mathbf{B}^T\mathbf{Z}_k\mathbf{A} \\ & (\mathbf{v}_1 + \mathbf{w}_1)^T\mathbf{B}\mathbf{C}_4 + \mathbf{w}_1^T\mathbf{A},\end{aligned}$$

$$\begin{aligned}\mathbf{v}_{2,k-1} = {} & \mathbf{C}_3^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{2,k})\tilde{\mathbf{C}}_1\hat{\mathbf{u}}_{k-1} + \mathbf{C}_3^T(\tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_1\hat{\mathbf{u}}_{k-1} + \\ & \mathbf{C}_3^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{2,k})\mathbf{C}_2\check{\mathbf{u}}_{k-1} + \mathbf{C}_3^T(\tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{M}}_k)\tilde{\mathbf{C}}_2\check{\mathbf{u}}_{k-1} + \\ & \mathbf{C}_3^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{2,k})\mathbf{c}_5 + \mathbf{C}_3^T(\tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{M}}_k)\mathbf{c}_5 + \\ & \mathbf{A}^T\mathbf{X}_k\mathbf{B}(\tilde{\mathbf{C}}_1\hat{\mathbf{u}}_{k-1} + \mathbf{C}_2\hat{\mathbf{u}}_{k-1} + \mathbf{c}_5) + \\ & \mathbf{A}^T\mathbf{Y}_{1,k}\mathbf{B}(\mathbf{C}_1\check{\mathbf{u}}_{k-1} + \tilde{\mathbf{C}}_2\check{\mathbf{u}}_{k-1} + \mathbf{c}_5) + \\ & \mathbf{C}_3^T\mathbf{B}^T(\mathbf{v}_2 + \mathbf{w}_2) + \mathbf{A}^T\mathbf{v}_2,\end{aligned}$$

$$\begin{aligned}\mathbf{w}_{2,k-1} = {} & \mathbf{C}_4^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{2,k})\tilde{\mathbf{C}}_1 + \mathbf{C}_4^T(\tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_1 + \\ & \mathbf{C}_4^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{2,k})\mathbf{C}_2 + \mathbf{C}_4^T(\tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{M}}_k)\tilde{\mathbf{C}}_2 + \\ & \mathbf{C}_4^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{2,k})\mathbf{c}_5 + \mathbf{C}_4^T(\tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{M}}_k)\mathbf{c}_5 + \\ & \mathbf{A}^T\mathbf{Y}_{2,k}\mathbf{B}(\tilde{\mathbf{C}}_1\hat{\mathbf{u}}_{k-1} + \mathbf{C}_2\hat{\mathbf{u}}_{k-1} + \mathbf{c}_5) + \\ & \mathbf{A}^T\mathbf{Z}_k\mathbf{B}(\mathbf{C}_1\check{\mathbf{u}}_{k-1} + \tilde{\mathbf{C}}_2\check{\mathbf{u}}_{k-1} + \mathbf{c}_5) + \\ & \mathbf{C}_4^T\mathbf{B}^T(\mathbf{v}_2 + \mathbf{w}_2) + \mathbf{A}^T\mathbf{w}_2,\end{aligned}$$

$$\begin{aligned}c_{k-1} = {} & \hat{\mathbf{u}}_{k-1}^T(\tilde{\mathbf{C}}_1^T\tilde{\mathbf{R}}_k\tilde{\mathbf{C}}_1 + \tilde{\mathbf{C}}_1^T\tilde{\mathbf{Y}}_{1,k}\mathbf{C}_1 + \mathbf{C}_1^T\tilde{\mathbf{Y}}_{2,k}\tilde{\mathbf{C}}_1 + \mathbf{C}_1^T\tilde{\mathbf{M}}_k\mathbf{C}_1)\hat{\mathbf{u}}_{k-1} + \\ & \hat{\mathbf{u}}_{k-1}^T(\tilde{\mathbf{C}}_1^T\tilde{\mathbf{R}}_k\mathbf{C}_2 + \tilde{\mathbf{C}}_1^T\tilde{\mathbf{Y}}_{1,k}\tilde{\mathbf{C}}_2 + \mathbf{C}_1^T\tilde{\mathbf{Y}}_{2,k}\mathbf{C}_2 + \mathbf{C}_1^T\tilde{\mathbf{M}}_k\tilde{\mathbf{C}}_2)\check{\mathbf{u}}_{k-1} + \\ & \check{\mathbf{u}}_{k-1}^T(\mathbf{C}_2^T\tilde{\mathbf{R}}_k\tilde{\mathbf{C}}_1 + \mathbf{C}_2^T\tilde{\mathbf{Y}}_{1,k}\mathbf{C}_1 + \tilde{\mathbf{C}}_2^T\tilde{\mathbf{Y}}_{2,k}\tilde{\mathbf{C}}_1 + \tilde{\mathbf{C}}_2^T\tilde{\mathbf{M}}_k\mathbf{C}_1)\hat{\mathbf{u}}_{k-1} + \\ & \check{\mathbf{u}}_{k-1}^T(\mathbf{C}_2^T\tilde{\mathbf{R}}_k\mathbf{C}_2 + \mathbf{C}_2^T\tilde{\mathbf{Y}}_{1,k}\tilde{\mathbf{C}}_2 + \tilde{\mathbf{C}}_2^T\tilde{\mathbf{Y}}_{2,k}\mathbf{C}_2 + \tilde{\mathbf{C}}_2^T\tilde{\mathbf{M}}_k\tilde{\mathbf{C}}_2)\check{\mathbf{u}}_{k-1} + \\ & (\mathbf{c}_5^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{2,k})\tilde{\mathbf{C}}_1 + \mathbf{c}_5^T(\tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{M}}_k)\mathbf{C}_1)\hat{\mathbf{u}}_{k-1} + (\mathbf{v}_1^T\mathbf{B}\tilde{\mathbf{C}}_1 + \mathbf{w}_1^T\mathbf{B}\mathbf{C}_1)\hat{\mathbf{u}}_{k-1} + \\ & (\mathbf{c}_5^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{2,k})\mathbf{C}_2 + \mathbf{c}_5^T(\tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{M}}_k)\tilde{\mathbf{C}}_2)\check{\mathbf{u}}_{k-1} + (\mathbf{v}_1^T\mathbf{B}\mathbf{C}_2 + \mathbf{w}_1^T\mathbf{B}\tilde{\mathbf{C}}_2)\check{\mathbf{u}}_{k-1} + \\ & \hat{\mathbf{u}}_{k-1}^T(\tilde{\mathbf{C}}_1^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k})\mathbf{c}_5 + \mathbf{C}_1^T(\tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{c}_5) + \hat{\mathbf{u}}_{k-1}^T(\tilde{\mathbf{C}}_1^T\mathbf{B}^T\mathbf{v}_2 + \mathbf{C}_1^T\mathbf{B}^T\mathbf{w}_2) + \\ & \check{\mathbf{u}}_{k-1}^T(\mathbf{C}_2^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k})\mathbf{c}_5 + \tilde{\mathbf{C}}_2^T(\tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{c}_5) + \check{\mathbf{u}}_{k-1}^T(\mathbf{C}_2^T\mathbf{B}^T\mathbf{v}_2 + \tilde{\mathbf{C}}_2^T\mathbf{B}^T\mathbf{w}_2) + \\ & \mathbf{c}_5^T(\tilde{\mathbf{R}}_k + \tilde{\mathbf{Y}}_{1,k} + \tilde{\mathbf{Y}}_{2,k} + \tilde{\mathbf{M}}_k)\mathbf{c}_5 + (\mathbf{v}_1^T + \mathbf{w}_1^T)\mathbf{B}\mathbf{c}_5 + \mathbf{c}_5^T\mathbf{B}^T(\mathbf{v}_2 + \mathbf{w}_2).\end{aligned}$$

C

Appendix to chapter 4

Pool Simulator

A pool model is necessary to simulate behavior of all balls on the table when executing a stroke. If the model is precise enough, results from the model can be transferred to the real world without further ado. the development of the pool model is described in this section. The section is split in two parts. In the first part, algebraic expressions of the physical model of a pool table are derived. As the expressions depend on a number of adjustable parameters, the second part coves the calibration in order to resemble the physical effects of the real pool table well.

Physical Model

To develop a precise pool model, the underlying physical effects are analyzed first. In total, there are five effects which can be considered individually. At the beginning of each player's move, a stroke occurs where the white ball is hit by the cue. After that, a usually short sliding phase is followed by a rather long rolling phase until the white ball comes to rest. In addition, collisions between two balls or a ball and the cushion may happen. Note that the effects described here are also deduced in different from in [5, 23, 61, 109, 208, 257, 302, 308, 313]

Stroke

As the cue hits the white ball, it transfers an impulse $\mathbf{p}$ in the x/y-plane to it. The contact point A between cue and ball is specified by the three parameters α accounting for

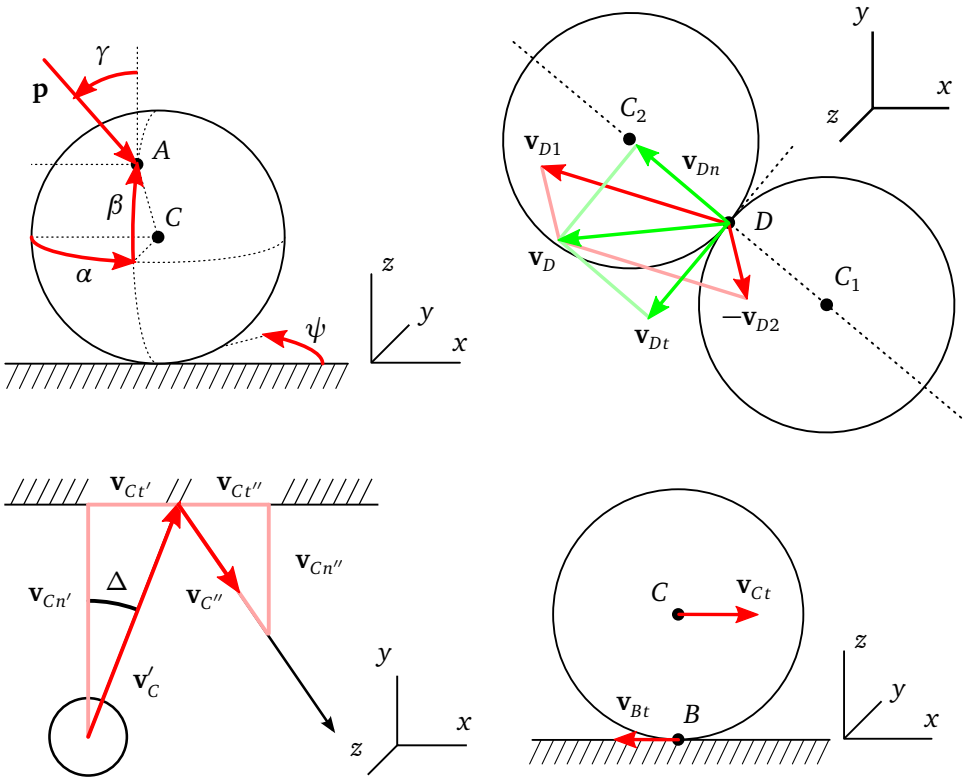


Figure C.1: Possible states of a pool game. Top left: Stroke. Top Right: Ball-ball collision. Bottom left: Ball-cushion collision. Bottom right: Rolling/sliding on the table

displacement of the cue along the y-axis causing side spin, β for displacement of the cue along the z-axis causing top spin and γ for the angle between z-axis and cue. Then the resulting COM velocity of the white ball $\mathbf{v}_C$ and the rotational velocity $\boldsymbol{\omega}_C$ is calculated as

$$\mathbf{v}_C = \frac{\|\mathbf{p}\|}{m}\begin{pmatrix}\sin\gamma\\0\\0\end{pmatrix},$$

$$\boldsymbol{\omega}_C = \frac{\|\mathbf{p}\|\|\mathbf{r}_{CA}\|}{m}\begin{pmatrix}\sin\alpha\cos\beta\cos\gamma\\-\cos\alpha\cos\beta\cos\gamma+\sin\beta\sin\gamma\\\sin\alpha\cos\beta\sin\gamma\end{pmatrix}.$$

In the equations above m denotes the mass of the ball and $\mathbf{r}_{CA}$ the vector from C to A. For arbitrary stroke directions on the table, $\mathbf{v}_C$ and $\boldsymbol{\omega}$ have to be rotated by an angle ψ around the z-axis.

Rolling on the Table

Differing from a ball sliding on a table, a rolling ball has no tangential speed $\mathbf{v}_{Bt}$ in the contact point B with the table. Thus it is $\mathbf{v}_{Bt} = \mathbf{0}$. The tangential rotational velocity $\boldsymbol{\omega}_t$ in the x/y-plane is then calculated as

$$\boldsymbol{\omega}_t = -\frac{1}{|\mathbf{r}_{CB}|^2}\,\mathbf{r}_{CB} \times \mathbf{v}_C.$$

While rolling on the table, the ball is slowed down by the rolling friction force $\mathbf{f}_{roll}$ acting in opposing direction of $\mathbf{v}_C$ as

$$\mathbf{f}_{roll} = -\frac{\mathbf{v}_C}{\|\mathbf{v}_C\|} mg\lambda_{ra} - \mathbf{v}_C mg\lambda_{rr},$$

with g for the gravitational constant, λ_{ra} for the velocity-independent rolling friction coefficient and λ_{rr} for the velocity-dependent rolling friction coefficient.

In case of a nonzero normal rotational velocity $\boldsymbol{\omega}_n$ along the z-axis, it will be unaffected by the rolling friction force $\mathbf{f}_{roll}$. Therefore [112] suggest to add another term for the normal angular acceleration $\boldsymbol{\alpha}_n$ depending on the radius ρ of the contact area between ball and table as

$$\boldsymbol{\alpha}_n = -\frac{\boldsymbol{\omega}_n}{|\boldsymbol{\omega}_n|}\frac{2}{3\Theta} mg\rho\lambda_{sa}, \tag{C.1}$$

depending on the inertia Θ of a ball. It can be calculated as $\Theta = 0.4mr_B$ with the parameter r_B as the radius of the ball.

Sliding on the Table

A ball sliding on the table is characterized by a translational velocity $\mathbf{v}_{Bt} \neq \mathbf{0}$. In this case the sliding friction force $\mathbf{f}_{slide}$ acting at the point B in opposing direction to $\mathbf{v}_{Bt}$ is calculated as

$$\mathbf{f}_{slide} = -\frac{\mathbf{v}_{Bt}}{|\mathbf{v}_{Bt}|} mg\lambda_{sa} - \mathbf{v}_{Bt} mg\lambda_{sr},$$

with λ_{sa} as the velocity-independent sliding friction coefficient and λ_{sr} as the velocity-dependent sliding friction coefficient. Similar to the previous case where the ball is rolling on the table, (C.1) holds.

Ball-Ball Collision

The collision between two balls is modeled as a partially elastic hit in normal direction. In case of a nonzero friction between the two balls, some impulse is transferred in tangential direction as well during the collision. If there are two balls with index 1 and 2 colliding, $\mathbf{v}_{D1}$ denotes the contact point velocity of ball 1 at the point D in a reference coordinate frame. The same accounts for $\mathbf{v}_{D2}$ and ball 2. The relative velocity between the two balls is then calculated as $\mathbf{v}_D = \mathbf{v}_{D2} - \mathbf{v}_{D1}$, consisting of a normal component $\mathbf{v}_{Dn}$ orthogonal to the

contact plane and a tangential component $\mathbf{v}_{Dt}$ along the contact plane. The transferred impulse $\mathbf{p}_n$ on ball 1 in normal direction is

$$\mathbf{p}_n = \frac{1 + e_{bb}}{2} m \mathbf{v}_{Dn},$$

where e_{bb} denotes the coefficient of restitution between two balls. The transferred impulse $\mathbf{p}_t$ in tangential direction on ball 1 is

$$\mathbf{p}_t = \|\mathbf{p}_n\| \lambda_{bb} \frac{\mathbf{v}_{Dt}}{\|\mathbf{v}_{Dt}\|}.$$

The resulting velocity $\mathbf{v}'_{C_1}$ of ball 1 after the hit is

$$\mathbf{v}'_{C_1} = \frac{\mathbf{p}_n + \mathbf{p}_t}{m} + \mathbf{v}_{C_1},$$

with $\mathbf{v}_{C_1}$ as the velocity before the hit. For a nonzero tangential impulse $\mathbf{p}_t$ the transferred angular momentum $\mathbf{L}_t$ on ball 1 as

$$\mathbf{L}_t = \mathbf{r}_{C_1 D} \times \mathbf{p}_t,$$

leads to the angular velocity $\boldsymbol{\omega}'_{C_1}$ after collision as

$$\boldsymbol{\omega}'_{C_1} = \frac{\mathbf{L}_t}{\Theta} + \boldsymbol{\omega}_{C_1}.$$

The resulting (angular) velocity of ball 2 is calculated accordingly.

Ball-Cushion Collision

A simplified model of the one presented in [112] is used to model the collision between ball and cushion. It assumes that the ball is rolling on the table immediately before and after cushion contact. If the ball has no sidespin, i.e. $\boldsymbol{\omega}_n = 0$, the contact with a cushion is modeled as a partially inelastic hit in normal direction as

$$\begin{aligned}
\mathbf{v}'_{Cn} &= -\mathbf{v}_{Cn} e_{bc}, \\
\mathbf{v}'_{Ct} &= \mathbf{v}_{Ct}, \\
\mathbf{v}'_{C} &= \frac{\mathbf{v}'_{Cn} + \mathbf{v}'_{Ct}}{\|\mathbf{v}'_{Cn} + \mathbf{v}'_{Ct}\|} \|\mathbf{v}_C\| \, \delta_v,
\end{aligned}$$

with the coefficient of restitution e_{bc} depending on both the velocity $\|\mathbf{v}_C\|$ before cushion contact and the angle Δ_C. The velocity $\mathbf{v}'_C$ after cushion contact is reduced by a factor δ_v relative to the velocity $\mathbf{v}_C$ before cushion contact.

Calibration

The developed pool model depends on a set of parameters which have to be determined through experiments. The following parameters are evaluated:

1. Ball mass m and radius r_B
2. Rolling friction coefficients λ_{ra} and λ_{rr}
3. Sliding friction coefficients λ_{sa} and λ_{sr}
4. Cushion parameters e_{bc} and δ_v

The parameters ρ, λ_{bb} and e_{bb} cannot be evaluated precisely enough due to an imperfect measurement setup or superimposed, more dominant effects. For those three parameters values based on [112] and [309] are used instead. Differing from purely event-based integration a fixed step integration size is used that can be adapted in case of an event (ball-ball or ball-cushion collision) [288].

Rolling friction coefficients

A ceiling mounted camera is used to track the position of a single ball on the table at a frame rate of 40Hz. As the ball rolls over the table, the rolling friction force $\mathbf{f}_{roll}$. By measuring the acceleration as the ball is slowed down, the rolling friction force can be deduced using Newton's second law. Fig. C shows the experimental results and the

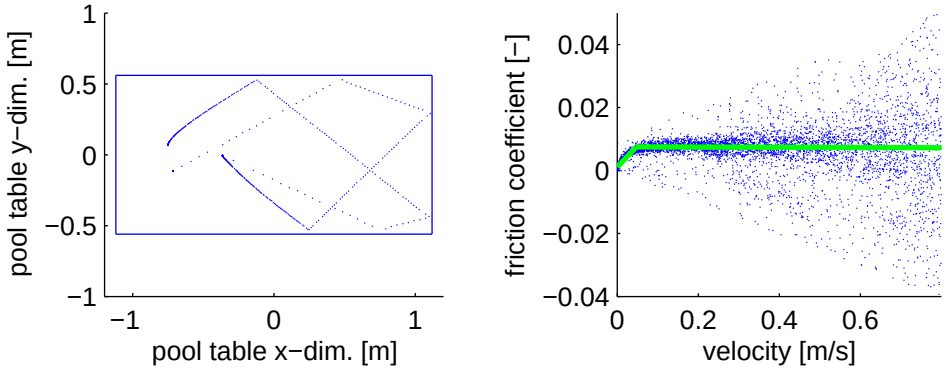


Figure C.2: Determination of the rolling friction coefficient. Left: Recorded trajectory by the ceiling-mounted camera for a single stroke. Right: Calculated rolling friction coefficient with the corresponding piecewise linear regression result. For better readability, only every tenth sampling point has been plotted

linear regression lines. For ball velocities $\|\mathbf{v}_C\| > 0.05\frac{m}{s}$ the rolling friction coefficient is nearly independent of the ball velocity. For ball velocities $\|\mathbf{v}_C\| \leq 0.05\frac{m}{s}$ a linear relation between the rolling friction coefficient and the ball velocity is observed. As shown in

the figure, outliers coming from the camera's limited resolution are removed for better regression results. The rolling friction force is calculated as

$$\lambda_{ra}(\mathbf{v}_C) = \begin{cases} 0.0011 & \text{for } |\mathbf{v}_C| \leq 0.0473, \\ 0.0075 & \text{for } |\mathbf{v}_C| > 0.0473, \end{cases}$$

$$\lambda_{rr}(\mathbf{v}_C) = \begin{cases} 0.1350 & \text{for } |\mathbf{v}_C| \leq 0.0473, \\ -0.0004 & \text{for } |\mathbf{v}_C| > 0.0473. \end{cases}$$

Sliding friction coefficients

A special rack with defined weight standing on three balls is designed to determine the sliding friction coefficients. The rack is pulled over the table by a linear axis with constant velocity. By fixing the balls to the rack, they are sliding over the table. A JR3 force/torque attached to the linear axis measures the force $\mathbf{f}_{slide}$ when the rack is moving. Multiple runs at different speeds between $0.01\frac{m}{s}$ and $1\frac{m}{s}$ are performed, see Fig. C.3. The rack

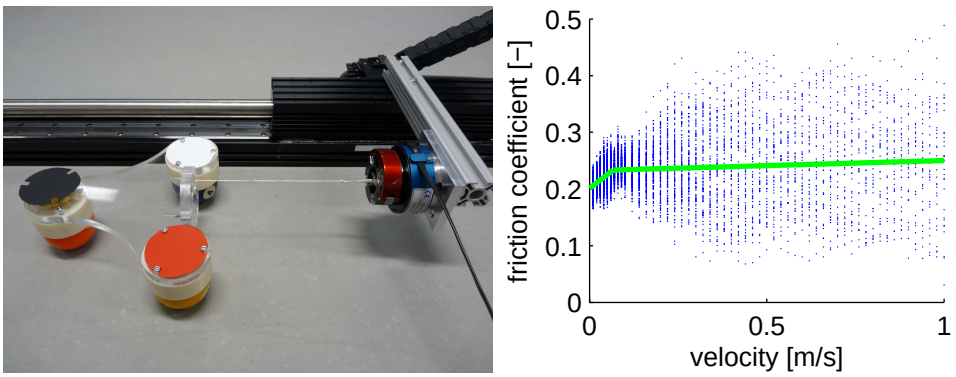


Figure C.3: Determination of the sliding friction coefficient. Left: Photo of the experimental setup. Right: Calculated sliding friction coefficient and piecewise linear regression lines

is attached to the linear axis through a slightly elastic string, causing small vibrations during the movement and resulting in noisy observation results. Similar to the rolling friction coefficient, the sliding friction coefficient is almost constant for large velocities and shows a nearly linear dependence for small velocities. A piecewise linear regression results in

$$\lambda_{sa}(\mathbf{v}_{Bt}) = \begin{cases} 0.2014 & \text{for } |\mathbf{v}_{Bt}| \leq 0.672, \\ 0.2322 & \text{for } |\mathbf{v}_{Bt}| > 0.672, \end{cases}$$

$$\lambda_{sr}(\mathbf{v}_{Bt}) = \begin{cases} 0.4763 & \text{for } |\mathbf{v}_{Bt}| \leq 0.672, \\ 0.0180 & \text{for } |\mathbf{v}_{Bt}| > 0.672. \end{cases}$$

Cushion contact coefficients

A single ball is shot at the cushion from multiple angles Δ_C and with varied velocity $\mathbf{v}_C$. Similar to the rolling friction coefficient, the parameters for the cushion contact are evaluated by tracking the trajectory of the ball on the table with a ceiling-mounted camera at 40Hz. Both the coefficient of restitution e_{bc} and the factor δ_v are determined as functions of the impact angle Δ_C and the velocity $\mathbf{v}_C$ before contact. Fig. C.4 displays the individual results with corresponding regression planes based on 1217 cushion contacts. The regression planes for e_{bc} and δ_v are

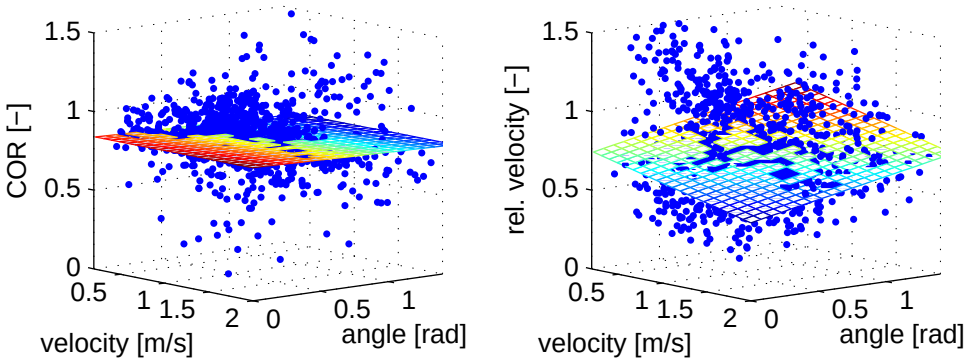


Figure C.4: Determination of the cushion parameters. Left: Coefficient of restitution e_{bc} of the cushion with corresponding regression plane for 1217 cushion contacts. Right: Relative velocity δ_v with calculated regression plane

$$e_{bc}(\mathbf{v}_C, \Delta) = 0.9331 - 0.0278|\mathbf{v}_C| - 0.1051\Delta,$$
$$\delta_v(\mathbf{v}_C, \Delta) = 0.7366 - 0.1094|\mathbf{v}_C| + 0.1698\Delta.$$

Subjective/objective difficulty

Two psychological experiments are performed in Sec. 4.6 to determine the subjective and objective difficulty of the human players. The parameters of both psychological experiments are shown in Tab. C.1 and C.1. In addition, Fig. C.5 and C.5 provide top-down and true-to-scale images of all scenarios for a better impression.

Experiment 1	d_1 green	d_2 green	θ_c green	d_1 red	d_2 red	θ_c red
Scen. 1	0.83m	0.70m	7.5°	0.11m	0.58m	7.7°
Scen. 2	1.37m	0.64m	0.7°	1.32m	0.49m	0.6°
Scen. 3	1.28m	0.38m	24.8°	0.75m	0.74m	25.2°
Scen. 4	0.57m	1.02m	24.7°	0.96m	0.64m	25.1°
Scen. 5	0.38m	0.34m	61.6°	0.67m	0.19m	61.3°
Scen. 6	0.38m	0.34m	61.4°	0.31m	0.57m	60.8°
Scen. 7	0.33m	0.17m	59.6°	0.33m	0.38m	11.8°
Scen. 8	0.14m	0.60m	46.5°	0.14m	0.37m	67.2°
Scen. 9	0.10m	0.56m	3.6°	0.10m	0.58m	17.4°
Scen. 10	0.89m	0.41m	28.4°	0.89m	0.36m	39.3°
Scen. 11	0.50m	1.03m	17.1°	0.50m	0.81m	33.5°
Scen. 12	1.40m	0.27m	32.4°	1.40m	0.28m	0.9°
Scen. 13	0.98m	1.10m	7.6°	0.89m	1.10m	24.1°
Scen. 14	0.47m	0.61m	14.7°	0.69m	0.61m	47.5°
Scen. 15	0.62m	0.11m	53.0°	0.24m	0.10m	74.6°
Scen. 16	1.23m	0.15m	5.4°	1.00m	0.15m	39.6°
Scen. 17	1.27m	0.31m	23.9°	1.36m	0.31m	50.9°
Scen. 18	0.79m	0.32m	1.4°	0.40m	0.31m	59.4°

Table C.1: Parameter overview of the first psychological experiment

Experiment 2	d_1	d_2	θ_c
Scen. 1	0.12m	0.10m	22.0°
Scen. 2	0.14m	0.12m	48.0°
Scen. 3	0.22m	0.12m	0.5°
Scen. 4	0.17m	0.35m	6.5°
Scen. 5	0.17m	0.35m	55.0°
Scen. 6	0.25m	0.46m	9.1°
Scen. 7	0.58m	0.23m	45.0°
Scen. 8	0.31m	0.68m	12.0°
Scen. 9	0.33m	0.56m	48.0°
Scen. 10	0.55m	0.52m	19.0°
Scen. 11	0.51m	0.75m	40.0°
Scen. 12	0.55m	0.94m	2.2°

Table C.2: Parameter overview of the second psychological experiment

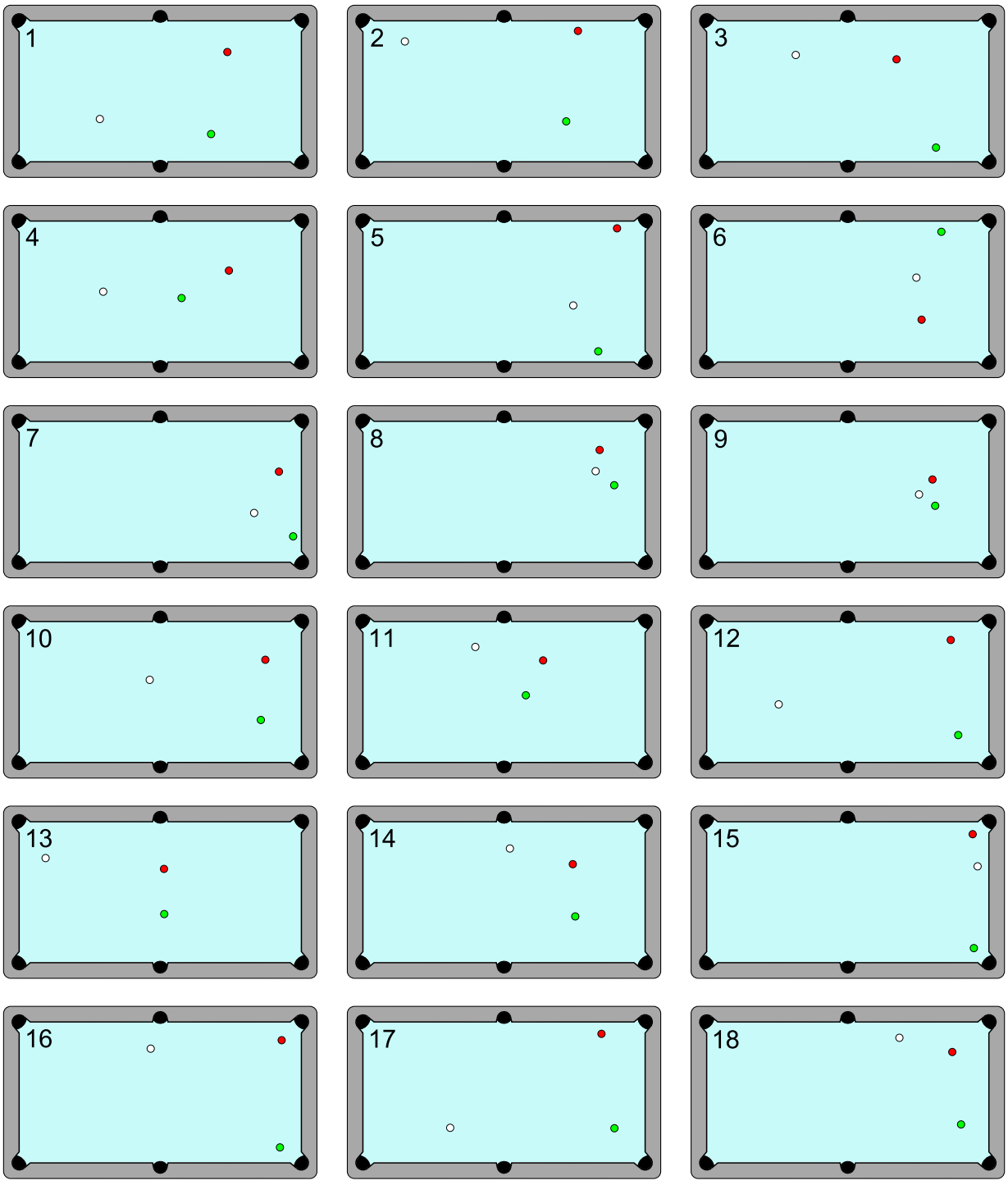


Figure C.5: Overview of the different scenarios of the first psychological experiment to determine the subjective difficulty

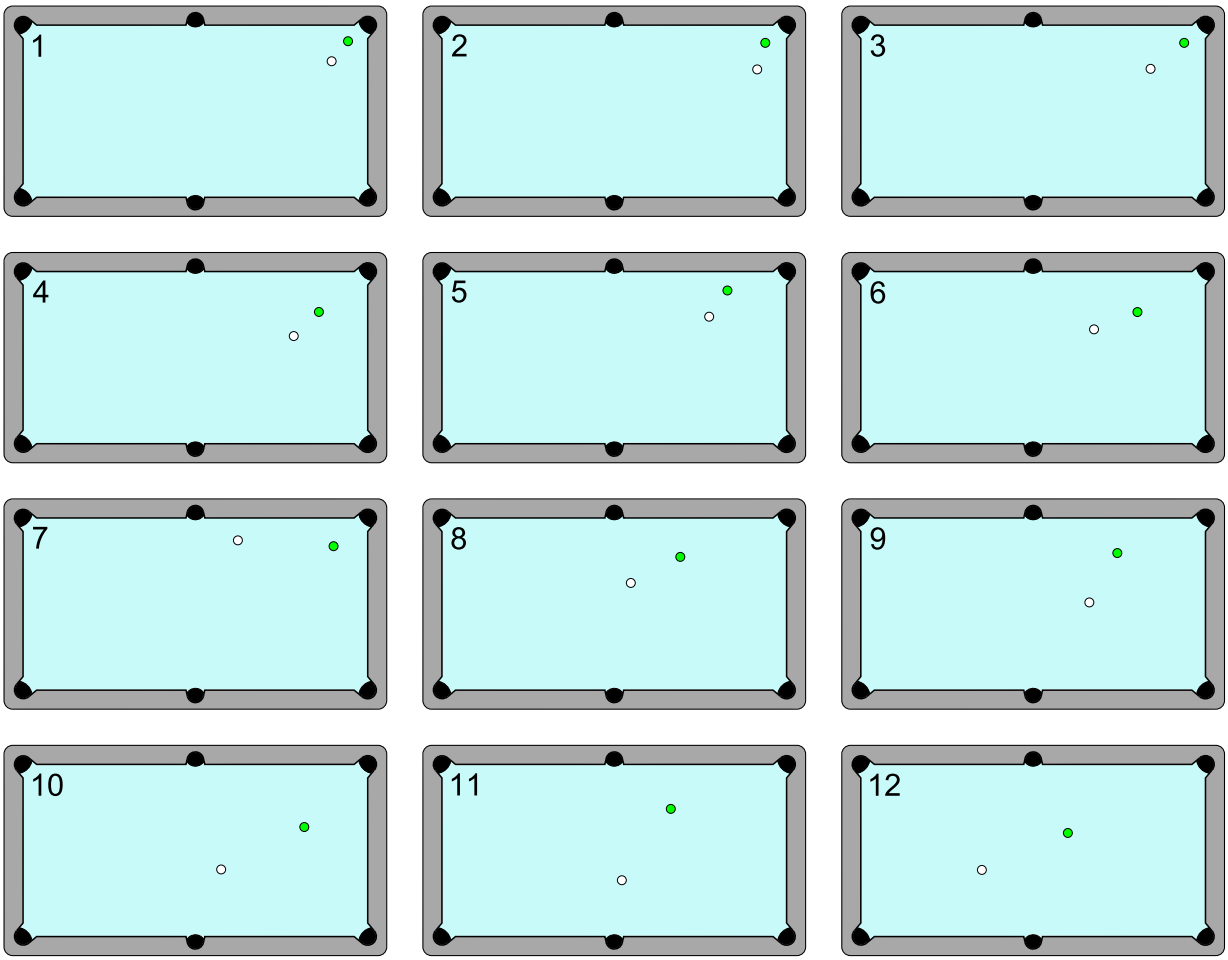


Figure C.6: Overview of the different scenarios of the second psychological experiment to determine the subjective difficulty

Bibliography

[1] euRobotics AISBL. *Robotics 2020 Multi-Annual Roadmap*. 2014. URL: http://www.eu-robotics.net/downloads/downloads/ (cited on p. 3).

[2] I. Ajzen. "The theory of planned behavior." In: *Organizational Behavior and Human Decision Processes* 50.2 (1991), pp. 179–211 (cited on p. 2).

[3] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. "Trajectories and keyframes for kinesthetic teaching: a human-robot interaction perspective." In: *ACM/IEEE International Conference on Human-Robot-Interaction*. 2012, pp. 391–398 (cited on p. 2).

[4] S. Albrecht, K. Ramirez-Amaro, F. Ruiz-Ugalde, D. Weikersdorfer, M. Leibold, M. Ulbrich, and M. Beetz. "Imitating human reaching motions using physically inspired optimization principles." In: *IEEE-RAS International Conference on Humanoid Robots*. 2011, pp. 602–607 (cited on p. 3).

[5] D. G. Alciatore. *The effects of cut angle, speed, and spin on object ball throw*. 2006. URL: http://billiards.colostate.edu/technical_proofs/new/TP_A-14.pdf (cited on p. 148).

[6] M. Alexa. "Differential coordinates for local mesh morphing and deformation." In: *The Visual Computer* 19.2 (2003), pp. 105–114 (cited on p. 13).

[7] M. Alexa, D. Cohen-Or, and D. Levin. "As-rigid-as-possible shape interpolation." In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 2000, pp. 157–164 (cited on p. 13).

[8] K. Alexopoulos, D. Mavrikios, M. Pappas, E. Ntelis, and G. Chryssolouris. "Multi-criteria upper-body human motion adaptation." In: *International Journal of Computer Integrated Manufacturing* 20.1 (2007), pp. 57–70 (cited on p. 3).

[9] M. E. Alian and S. B. Shouraki. "A Fuzzy Pool Player Robot with Learning Ability." In: *WSEAS Transactions on Electronics* 2, no. 1 (2004), pp. 422–426 (cited on p. 115).

[10] M. E. Alian, S. B. Shouraki, M. T. M. Shalmani, P. Karimian, and P. Sabzmeydani. "Roboshark: a gantry pool player robot." In: *International Symposium on Robotics*. 2004 (cited on p. 115).

[11] D. Althoff, O. Kourakos, M. Lawitzky, A. Mörtl, M. Rambow, F. Rohrmüller, D. Brscic, D. Wollherr, S. Hirche, and M. Buss. “An Architecture for Real-Time Control in Multi-Robot Systems.” In: *Cognitive Systems Monographs*. Springer, 2009, pp. 43–52 (cited on p. 133).

[12] N. Ando, T. Suehiro, and T. Kotoku. “A Software Platform for Component Based RT-System Development: OpenRTM-Aist.” In: *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. 2008, pp. 87–98 (cited on pp. 54, 57).

[13] C. Archibald, A. Altman, and Y. Shoham. “Analysis of a Winning Computational Billiards Player.” In: *International Joint Conference on Artificial Intelligence*. 2009, pp. 1377–1382 (cited on pp. 115, 130, 134, 136).

[14] C. Archibald and Y. Shoham. “Modeling billiards games.” In: *International Conference on Autonomous Agents and Multiagent Systems*. 2009, pp. 193–199 (cited on p. 115).

[15] A. Artale and E. Franconi. “A Temporal Description Logic for Reasoning about Actions and Plans.” In: *Journal of Artificial Intelligence Research (JAIR)* 9 (1998), pp. 463–506 (cited on p. 114).

[16] K. S. Arun, T. S. Huang, and S. D. Blostein. “Least-Squares Fitting of Two 3-D Point Sets.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9.5 (1987), pp. 698–700 (cited on pp. 20, 28).

[17] K. Åström. *Adaptive Control*. Pearson Education, 2006 (cited on p. 80).

[18] X. Bai and L. J. Latecki. “Path similarity skeleton graph matching.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.7 (2008) (cited on p. 63).

[19] C. Baral, M. Gelfond, and A. Provetti. “Representing Actions: Laws, Observations and Hypotheses.” In: *Journal of Logic and Algebraic Programming* 31.1-3 (1997), pp. 201–243 (cited on p. 114).

[20] J. Barraquand, B. Langlas, and J. C. Latombe. “Numerical Potential Field Techniques for Robot Path Planning.” In: *IEEE Transactions on Systems, Man, and Cybernetics* 22 (1992), pp. 224–241 (cited on pp. 11, 13).

[21] A. G. Barto, S. J. Bradtke, and S. P. Singh. “Learning to act using real-time dynamic programming.” In: *Artificial Intelligence* 72.1-2 (1995), pp. 81–138 (cited on p. 123).

[22] F. I. Bashir, A. A. Khokhar, and D. Schonfeld. “Segmented trajectory based indexing and retrieval of video data.” In: *IEEE International Conference on Image Processing*. 2003, pp. 623–626 (cited on p. 63).

[23] J. H. Bayes and W. T. Scott. “Billard-Ball Collision Experiment.” In: *American Journal of Physics* 31, no.3 (1963), pp. 197–200 (cited on p. 148).

[24] M. Beetz, D. Jain, L. Mösenlechner, and M. Tenorth. “Towards performing everyday manipulation activities.” In: *Robotics and Autonomous Systems* 58.9 (2010), pp. 1085–1095 (cited on p. 63).

[25] R. Bellman. “A Markovian Decision Process.” In: *Indiana University Mathematics Journal* 6 (1957), pp. 679–684 (cited on p. 115).

[26] T. J. M. Bench-Capon and K. Atkinson. “Action-State Semantics for Practical Reasoning.” In: *AAAI Fall Symposium: The Uses of Computational Argumentation*. Vol. FS-09-06. 2009 (cited on p. 3).

[27] M. Benzi. “Preconditioning Techniques for Large Linear Systems: A Survey.” In: *Journal of Computational Physics* 182.2 (2002), pp. 418–477 (cited on p. 23).

[28] J. van den Berg, D. Wilkie, S. Guy, M. Niethammer, and D. Manocha. “LQG-obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty.” In: *International Conference on Robotics and Automation*. 2012, pp. 346–353 (cited on p. 63).

[29] N. Bernstein. *The coordination and regulation of movement*. Pergamon Press, 1967 (cited on p. 10).

[30] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. 2nd. Athena Scientific, 2000 (cited on p. 115).

[31] D. P. Bertsekas and J. N. Tsitsiklis. “An Analysis of Stochastic Shortest Path Problems.” In: *Mathematics of Operations Research* 16.3 (1991), pp. 580–595 (cited on p. 118).

[32] E. Bicho, W. Erlhagen, L. Louro, and E. C. e Silva. “Neuro-cognitive mechanisms of decision making in joint action: A human-robot interaction study.” In: *Human Movement Science* 30.5 (2011), pp. 846 –868 (cited on p. 114).

[33] A. Billard, S. Calinon, and F. Guenter. “Discriminative and Adaptive Imitation in Uni-Manual and Bi-Manual Tasks.” In: *Robotics and Autonomous Systems* 54.5 (2006), pp. 370–384 (cited on pp. 71, 90).

[34] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. “Robot Programming by Demonstration.” In: *Springer Handbook of Robotics*. 2008, pp. 1371–1394 (cited on pp. 10, 85).

[35] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. “Time-optimal Control of Robotic Manipulators along Specified Paths.” In: *International Journal of Robotics Research* 4.3 (1985), pp. 3–17 (cited on p. 63).

[36] G. Bornstein. *Intergroup conflict: Individual, group and collective interests*. Tech. rep. dp297. The Center for the Study of Rationality, Hebrew University, Jerusalem, 2002 (cited on p. 114).

[37] M. Botsch and L. Kobbelt. “A remeshing approach to multiresolution modeling.” In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 2004, pp. 185–192 (cited on p. 29).

[38] M. Botsch and L. Kobbelt. “An intuitive framework for real-time freeform modeling.” In: *ACM Transactions on Graphics* 23.3 (2004), pp. 630–634 (cited on p. 13).

[39] O. Brock and O. Khatib. “Elastic Strips: A Framework for Motion Generation in Human Environments.” In: *International Journal of Robotics Research* 21.12 (2002), pp. 1031–1052 (cited on pp. 11, 38).

[40] D. Brscic, M. Eggers, F. Rohrmüller, O. Kourakos, S. Sosnowski, D. Althoff, M. Lawitzky, A. Mörtl, M. Rambow, V. Koropouli, J. R. M. Hernández, X. Zang, W. Wang, D. Wollherr, K. Kühnlenz, C. Mayer, T. Kruse, A. Kirsch, J. Blume, A. Bannat, T. Rehrl, F. Wallhoff, T. Lorenz, P. Basili, C. Lenz, T. Röder, G. Panin, W. Maier, S. Hirche, M. Buss, M. Beetz, B. Radig, A. Schubö, S. Glasauer, A. Knoll, and E. Steinbach. *Multi Joint Action in CoTeSys - setup and challenges*. 2010 (cited on p. 133).

[41] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. *Detecting Commuting Patterns by Clustering Subtrajectories*. 2008 (cited on p. 63).

[42] M. Buchin, A. Driemel, M. van Kreveld, and V. Sacristán. "An algorithmic framework for segmenting trajectories based on spatio-temporal criteria." In: *ACM Special Interest Group on Spatial Information*. 2010, pp. 202–211 (cited on p. 63).

[43] E Burdet, K. Tee, I Mareels, T. Milner, C. Chew, D. Franklin, R Osu, and M Kawato. "Stability and motor adaptation in human arm movements." In: *Biological Cybernetics* 94 (2006), pp. 20–32 (cited on p. 2).

[44] A. N. Burnetas and M. N. Katehakis. "Optimal Adaptive Policies for Markov Decision Processes." In: *Mathematics of Operations Research* 22.1 (1997), pp. 222–255 (cited on p. 115).

[45] S. Calderara, R. Cucchiara, and A. Prati. "A Dynamic Programming Technique for Classifying Trajectories." In: *International Conference on Image Analysis and Processing*. 2007, pp. 137–142 (cited on p. 63).

[46] S. Calinon. *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009 (cited on pp. 12, 105).

[47] S. Calinon, I. Sardellitti, and D. G. Caldwell. "Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 249–254 (cited on p. 11).

[48] S. Calinon and A. Billard. "Statistical Learning by Imitation of Competing Constraints in Joint Space and Task Space." In: *Advanced Robotics* 23.15 (2009), pp. 2059–2076 (cited on p. 86).

[49] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. Billard. "Learning and Reproduction of Gestures by Imitation." In: *IEEE Robotics and Automation Magazine* 17.2 (2010), pp. 44–54 (cited on pp. 11, 13).

[50] S. Calinon, F. Guenter, and A. Billard. "On learning the statistical representation of a task and generalizing it to various contexts." In: *IEEE International Conference on Robotics and Automation*. 2006, pp. 2978–2983 (cited on p. 11).

[51] S. Calinon, F. Guenter, and A. Billard. "On Learning, Representing and Generalizing a Task in a Humanoid Robot." In: *IEEE Transactions on Systems, Man, and Cybernetics* 37.2 (2007), pp. 286–298 (cited on p. 11).

[52] M. Campbell, A. J. Hoane Jr., and F.-h. Hsu. "Deep Blue." In: *Artificial Intelligence* 134.1-2 (2002), pp. 57–83 (cited on p. 115).

[53] S. W. S. Chang. "Automating Skills using a Robot Snooker Player." PhD thesis. Bristol University, 1994 (cited on p. 115).

[54] W. Cheung and S. Sang. “Automating Skills Using a Robot Snooker Player.” PhD thesis. 1994 (cited on p. 115).

[55] B. Chu, K. Jung, C.-S. Han, and D. Hong. “A survey of climbing robots: Locomotion and adhesion.” In: *International Journal of Precision Engineering and Manufacturing* 11.4 (2010), pp. 633–647 (cited on p. 10).

[56] S. Chua, E. Wong, A. Tan, and V. Koo. “Decision Algorithm for Pool Using Fuzzy System.” In: *International Conference on Artificial Intelligence and Applications in Engineering and Technology*. 2002, pp. 370–375 (cited on p. 120).

[57] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997 (cited on p. 18).

[58] W. Chung, L.-C. Fu, and S.-H. Hsu. “Motion Control.” In: *Springer Handbook of Robotics*. 2008, pp. 133–159 (cited on p. 83).

[59] J. M. Cioffi and T. Kailath. “Fast, recursive-least-squares transversal filters for adaptive filtering.” In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32 (2 1984), pp. 304–337 (cited on p. 80).

[60] A. Croitoru, P. Agouris, and A. Stefanidis. “3D trajectory matching by pose normalization.” In: *ACM International Conference on Information and Knowledge Management*. 2005, pp. 153–162 (cited on pp. 63, 110).

[61] R. Cross. “Cue and ball deflection in billiards.” In: *American Journal of Physics* 76, no.3 (2007), pp. 205–212 (cited on p. 148).

[62] M. R. Cutkosky and K. Sangbae. *Controllable and directional dry adhesive structure*. WO Patent App. PCT/US2007/008,749. 2008 (cited on p. 2).

[63] K. Dautenhahn. “Methodology & Themes of Human-Robot Interaction: A Growing Research Field.” In: *International Journal of Advanced Robotic Systems* 4.1 (2007), pp. 103–108 (cited on p. 114).

[64] A. De Santis. “Modelling and Control for Human-Robot Interaction.” PhD thesis. Università degli Studi di Napoli Federico II, 2007 (cited on p. 2).

[65] J. De Schutter. “Invariant description of rigid body motion trajectories.” In: *ASME Journal of Mechanisms and Robotics* 2 (2010), pp. 1–9 (cited on p. 63).

[66] Y. Demiris. “Prediction of intent in robotics and multi-agent systems.” In: *Cognitive Processing* 8.3 (2007), pp. 151–158 (cited on p. 114).

[67] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. “Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow.” In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 1999, pp. 317–324 (cited on p. 18).

[68] R. Diankov. “Automated Construction of Robotic Manipulation Programs.” PhD thesis. Carnegie Mellon University, Robotics Institute, 2010 (cited on pp. 57, 102).

[69] R. Diankov and J. Kuffner. *OpenRAVE: A Planning Architecture for Autonomous Robotics*. Tech. rep. CMU-RI-TR-08-34. Robotics Institute, 2008 (cited on p. 54).

[70] U. Dierkes, S. Hildebrandt, and F. Sauvigny. *Minimal surfaces*. 1 ; 339. Springer, 2010, XV, 688 S. (Cited on p. 13).

[71] H. Ding, G. Trajcevski, and P. Scheuermann. *Efficient Similarity Join of Large Sets of Moving Object Trajectories*. 2008 (cited on p. 63).

[72] M. Do, P. Azad, T. Asfour, and R. Dillmann. "Imitation of Human Motion on a Humanoid Robot using Nonlinear Optimization." In: *IEEE/RAS International Conference on Humanoid Robots*. 2008, pp. 545–552 (cited on p. 2).

[73] M. P. Do-Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976 (cited on p. 19).

[74] P. Dorato and A. H. Levis. "Optimal linear regulators: The discrete-time case." In: *Automatic Control, IEEE Transactions on* 16.6 (1971), pp. 613–620 (cited on pp. 76, 82).

[75] A. D'Souza, S. Vijayakumar, and S. Schaal. "Learning inverse kinematics." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 1. 2001, 298–303 vol.1 (cited on p. 62).

[76] J.-P. Dussault and J.-F. Landry. "Optimization of a Billiard Player - Tactical Play." In: *Computers and Games*. 2006 (cited on pp. 120, 136).

[77] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. "Multiresolution analysis of arbitrary meshes." In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 1995, pp. 173–182 (cited on p. 28).

[78] M. Egerstedt, S. Martini, M. Cao, K. Camlibel, and A. Bicchi. "Interacting with networks: How does structure relate to controllability in single-leader, consensus networks?" In: *Control Systems Magazine* 32.4 (2012), pp. 66 –73 (cited on p. 24).

[79] F. Enner, D. Rollinson, and H. Choset. "Simplified motion modeling for snake robots." In: *IEEE International Conference on Robotics and Automation*. 2012, pp. 4216–4221 (cited on p. 2).

[80] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard. "Imitation learning with generalized task descriptions." In: *IEEE International Conference on Robotics and Automation*. 2009, pp. 3968–3974 (cited on p. 62).

[81] W. Erlhagen, A. Mukovskiy, E. Bicho, G. Panin, C. Kiss, A. Knoll, H. van Schie, and H. Bekkering. "Action understanding and imitation learning in a robot-human task." In: *International Conference on Artificial Neural Networks*. Vol. 1. 2005, pp. 261–268 (cited on p. 114).

[82] J. Ernesti, L. Righetti, M. Do, T. Asfour, and S. Schaal. "Encoding of Periodic and their Transient Motions by a Single Dynamic Movement Primitive." In: *IEEE-RAS International Conference on Humanoid Robots*. 2012, pp. 57–64 (cited on p. 11).

[83] O. Etzioni, S. Hanks, D. S. Weld, D. Draper, N. Lesh, and M. Williamson. "An Approach to Planning with Incomplete Information." In: *Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 1992, pp. 115–125 (cited on p. 114).

[84] R. Featherstone and D. E. Orin. "Robot Dynamics: Equations and Algorithms." In: *IEEE International Conference on Robotics and Automation*. 2000, pp. 826–834 (cited on p. 63).

[85] D. Feil-Seifer and M. J. Mataric. "Defining Socially Assistive Robotics." In: *International Conference on Rehabilitation Robotics*. 2005, pp. 465–468 (cited on p. 114).

[86] P. F. Felzenszwalb. "Hierarchical Matching of Deformable Shapes." In: *Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8 (cited on pp. 96, 97, 111).

[87] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. "A Clustering Technique for the Identification of Piecewise Affine Systems." In: *Hybrid Systems: Computation and Control*. Vol. 2034. 2001, pp. 218–231 (cited on p. 63).

[88] R. Findeisen, L. Imsland, F. Allgower, and B. A. Foss. "State and output feedback nonlinear model predictive control: An overview." In: *European Journal of Control* 9.2 (2003), pp. 190–206 (cited on p. 63).

[89] P. Fiorini and Z. Shiller. "Motion Planning in Dynamic Environments Using Velocity Obstacles." In: *International Journal of Robotics Research* 17.7 (1998), pp. 760–772 (cited on p. 11).

[90] T. Flash and N. Hogan. "The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model." In: *Journal of Neuroscience* 5.7 (1985), pp. 1688–1703 (cited on p. 26).

[91] M. S. Floater. "Mean Value Coordinates." In: *Computer Aided Geometric Design* 20.1 (2003), pp. 19–27 (cited on p. 13).

[92] T. Fong, I. Nourbakhsh, and K. Dautenhahn. "A survey of socially interactive robots." In: *Robotics and Autonomous Systems* 42.3-4 (2003), pp. 143 –166 (cited on p. 10).

[93] D. Forte, A. Gams, J. Morimoto, and A. Ude. "On-line motion synthesis and adaptation using a trajectory database." In: *Robotics and Autonomous Systems* 60.10 (2012), pp. 1327–1339 (cited on p. 12).

[94] B. Friedlander. "Lattice filters for adaptive processing." In: *Proceedings of the IEEE* 70.8 (1982), pp. 829–867 (cited on p. 80).

[95] H. Friedrich, S. Münch, R. Dillmann, S. Bocionek, and M. Sassin. "Robot programming by Demonstration (RPD): Supporting the induction by human interaction." In: *Machine Learning* 23.2-3 (1996), pp. 163–189 (cited on p. 10).

[96] J. García-Diego. *Juanelo Turriano, Charles V's Clockmaker: The Man and His Legend*. Science History Publications, 1986 (cited on p. 1).

[97] M. Gelfond and V. Lifschitz. "Representing Actions in Extended Logic Programming." In: *Joint International Conference and Symposium on Logic Programming*. 1992, pp. 559–573 (cited on p. 3).

[98] R. Geraerts and M. Overmars. "A Comparative Study of Probabilistic Roadmap Planners." In: *Algorithmic Foundations of Robotics V*. Vol. 7. Springer Berlin Heidelberg, 2004, pp. 43–57 (cited on p. 3).

[99] M. J. Gielniak and A. L. Thomaz. "Generating anticipation in robot motion." In: *IEEE International Workshop on Robots and Human Interactive Communications*. 2011, pp. 449–454 (cited on p. 4).

[100] E. L. Glassman and R. Tedrake. “A quadratic regulator-based heuristic for rapidly exploring state space.” In: *IEEE International Conference on Robotics and Automation*. 2010, pp. 5021–5028 (cited on p. 63).

[101] M. Gleicher and P. Litwinowicz. “Constraint-based motion adaptation.” In: 1998, pp. 65–94 (cited on p. 12).

[102] M. Goebl and G. Färber. “A Real-Timeime-capable Hard- and Software Architecture for Joint Image and Knowledge Processing in Cognitive Automobiles.” In: *IEEE Intelligent Vehicles Symposium*. 2007 (cited on p. 133).

[103] V. M. Gonçalves, L. C. A. Pimenta, C. A. Maia, B. C. O. Dutra, and G. A. S. Pereira. “Vector Fields for Robot Navigation Along Time-Varying Curves in n -Dimensions.” In: *IEEE Transactions on Robotics* 26.4 (2010), pp. 647–659 (cited on p. 63).

[104] M. A. Goodrich and A. C. Schultz. “Human-robot interaction: a survey.” In: *Foundations and trends in human computer interaction* 1.3 (2007), pp. 203–275 (cited on p. 114).

[105] M. Greenspan, J. Lam, M. Godard, I. Zaidi, S. Jordan, W. Leckie, K. Anderson, and D. Dupuis. “Toward a Competitive Pool Playing Robot: Is Computational Intelligence Needed to Play Robotic Pool?” In: *IEEE Symposium on Computational Intelligence and Games*. 2007, pp. 380 –388 (cited on p. 115).

[106] *Discrete differential geometry: an applied introduction*. ACM Special Interest Group on Graphics and Interactive Techniques Courses Notes. 2006 (cited on p. 13).

[107] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. “Style-based inverse kinematics.” In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 2004, pp. 522–531 (cited on p. 62).

[108] F. Guenter, M. Hersch, S. Calinon, and A. Billard. “Reinforcement Learning for Imitating Constrained Reaching Movements.” In: *Advanced Robotics* 21.13 (2007), pp. 1521–1544 (cited on p. 3).

[109] D. Gugan. “Inelastic collision and the Hertz theory of impact.” In: *American Journal of Physics* 68, no.10 (1999), pp. 920–924 (cited on p. 148).

[110] E. Guigon, P. Baraduc, and M. Desmurget. “Computational motor control: redundancy and invariance.” In: *Journal of Neurophysiology* 97.1 (2007), pp. 331–47 (cited on p. 10).

[111] I. Guskov, W. Sweldens, and P. Schröder. “Multiresolution Signal Processing for Meshes.” In: *Special Interest Group on Graphics and Interactive Techniques*. 1999, pp. 325–334 (cited on p. 28).

[112] I. Han. “Dynamics in Carom and Three Cushion Billiards.” In: *Journal of Mechanical Science and Technology* 19, no.4 (2005), pp. 976–984 (cited on pp. 136, 150–152).

[113] E. A. Hansen and S. Zilberstein. “LAO*: A heuristic search algorithm that finds solutions with loops.” In: *Artificial Intelligence* 129.1-2 (2001), pp. 35–62 (cited on p. 123).

[114] G. A. Hansen, R. W. Douglass, and A. Zardecki. *Mesh Enhancement: Selected Elliptic Methods, Foundations, and Applications*. Imperial College Press, 2005 (cited on p. 13).

[115] S. Haykin. *Adaptive filter theory*. 4th. Prentice Hall, 2002 (cited on p. 80).

[116] M. Hersch, F. Guenter, S. Calinon, and A. Billard. "Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations." In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1463–1467 (cited on p. 2).

[117] L. Hilario, N. Montés, M. C. Mora, and A. Falcó. "Real-time Bézier Trajectory Deformation for Potential Fields planning methods." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 1567–1572 (cited on p. 11).

[118] K. Hildebrandt and K. Polthier. "On approximation of the Laplace-Beltrami operator and the Willmore energy of surfaces." In: *Computer Graphics Forum* 30.5 (2011), pp. 1513–1520 (cited on p. 18).

[119] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. "The Development of Honda Humanoid Robot." In: *IEEE International Conference on Robotics and Automation*. 1998, pp. 1321–1326 (cited on p. 2).

[120] E. S. L. Ho, T. Komura, and C.-L. Tai. "Spatial relationship preserving character motion adaptation." In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 2010, 33:1–33:8 (cited on pp. 90, 111).

[121] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal. "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance." In: *IEEE International Conference on Robotics and Automation*. 2009, pp. 2587–2592 (cited on p. 11).

[122] D. Hsu, J.-C. Latombe, and H. Kurniawati. "On the Probabilistic Foundations of Probabilistic Roadmap Planning." In: *International Society of Root Research*. Vol. 28. 2005, pp. 83–97 (cited on p. 11).

[123] M. Huber, A. Kupferberg, C. Lenz, A. Knoll, T. Brandt, and S. Glasauer. "Spatiotemporal Movement Planning and Rapid Adaptation for Manual Interaction." In: *PLoS ONE* 8.5 (2013), e64982 (cited on p. 114).

[124] T. Igarashi, T. Moscovich, and J. F. Hughes. "As-rigid-as-possible shape manipulation." In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 2005, pp. 1134–1141 (cited on p. 20).

[125] S. Ikemoto, H. B. Amor, T. Minato, H. Ishiguro, and B. Jung. "Physical interaction learning: Behavior adaptation in cooperative human-robot tasks involving physical contact." In: *IEEE International Symposium on Robot and Human Interactive Communication*. 2009, pp. 504–509 (cited on p. 3).

[126] S. Ikemoto, H. B. Amor, T. Minato, B. Jung, and H. Ishiguro. "Physical Human-Robot Interaction: Mutual Learning and Adaptation." In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 24–35 (cited on p. 114).

[127] L. Jaillet and J. Porta. "Path Planning Under Kinematic Constraints by Rapidly Exploring Manifolds." In: *IEEE Transactions on Robotics* 29.1 (2013), pp. 105–117 (cited on p. 11).

[128] N. Jarrassé, T. Charalambous, and E. Burdet. "A framework to describe, analyze and generate interactive motor behaviors." In: *PloS one* 7.11 (2012), e49945–e49945 (cited on p. 114).

[129] R. N. Jazar. *Theory of Applied Robotics: Kinematics, Dynamics, and Control*. Springer, 2007 (cited on p. 62).

[130] J. Jeon, S. Karaman, and E. Frazzoli. "Anytime Computation of Time-Optimal Off-Road Vehicle Maneuvers using the RRT*." In: *IEEE Conference on Decision and Control*. 2011, pp. 3276–3282 (cited on p. 59).

[131] N. Jetchev. "Learning representations from motion trajectories: analysis and applications to robot planning and control." PhD thesis. FU Berlin, 2012 (cited on p. 62).

[132] N. Jetchev and M. Toussaint. "Task Space Retrieval Using Inverse Feedback Control." In: *International Conference on Machine Learning*. 2011, pp. 449–456 (cited on p. 62).

[133] M. Johansson. *Piecewise Linear Control Systems*. Springer, 2003 (cited on p. 63).

[134] M. Jordan and A. Perez. *Optimal Bidirectional Rapidly-Exploring Random Trees*. Tech. rep. MIT, 2013 (cited on p. 11).

[135] M. R. Jung and N. L. Badler. *Generating Human Motion By Symbolic Reasoning*. Tech. rep. University of Pennsylvania, 1992 (cited on p. 63).

[136] M. W. Kadous. "Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series." PhD thesis. University of New South Wales, 2002 (cited on pp. 107, 110).

[137] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. "STOMP: Stochastic Trajectory Optimization for Motion Planning." In: *IEEE International Conference on Robotics and Automation*. 2011 (cited on p. 11).

[138] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. "Humanoid robot HRP-2." In: *IEEE International Conference on Robotics and Automation*. Vol. 2. 2004, pp. 1083–1090 (cited on p. 2).

[139] K. Kaneko, F. Kanehiro, M. Morisawa, K. Akachi, G. Miyamori, A. Hayashi, and N. Kanehira. "Humanoid robot HRP-4 - Humanoid robotics platform with lightweight and slim body -." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 4400–4407 (cited on pp. 2, 54).

[140] S. Karaman and E. Frazzoli. "Incremental Sampling-based Algorithms for Optimal Motion Planning." In: *Robotics: Science and Systems*. 2010 (cited on p. 11).

[141] S. Karaman and E. Frazzoli. "Optimal Kinodynamic Motion Planning using Incremental Sampling-based Methods." In: *IEEE Conference on Decision and Control*. 2010 (cited on p. 60).

[142] S. Karaman and E. Frazzoli. "Sampling-based Algorithms for Optimal Motion Planning." In: *International Journal of Robotics Research* 30.7 (2011), pp. 846–894 (cited on pp. 11, 40, 42).

[143] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller. "Real-time Motion Planning using the RRT*." In: *IEEE International Conference on Robotics and Automation*. 2011 (cited on p. 59).

[144] S. Karaman and E. Frazzoli. “Sampling-based optimal motion planning for non-holonomic dynamical systems.” In: *IEEE International Conference on Robotics and Automation*. 2013, pp. 5041–5047 (cited on p. 59).

[145] Z. Karni and C. Gotsman. “Spectral compression of mesh geometry.” In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 2000, pp. 279–286 (cited on p. 13).

[146] Z. Kasap and N. Magnenat-Thalmann. “Interacting with Emotion and Memory Enabled Virtual Characters and Social Robots.” In: *Modeling Machine Emotions for Realizing Intelligence*. Vol. 1. Smart Innovation, Systems and Technologies. Springer Berlin Heidelberg, 2010, pp. 209–224 (cited on p. 114).

[147] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars. *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*. Tech. rep. 1994 (cited on p. 11).

[148] E. J. Keogh and M. J. Pazzani. “Scaling up Dynamic Time Warping for Datamining Applications.” In: *ACM Special Interest Group on Knowledge Discovery and Data Mining*. 2000, pp. 285–289 (cited on pp. 63, 105, 110).

[149] S.-M. Khansari-Zadeh and A. Billard. “A dynamical system approach to realtime obstacle avoidance.” In: *Autonomous Robots* 32 (4 2012), pp. 433–454 (cited on p. 63).

[150] M. Khatib, H. Jaouni, R. Chatila, and J. Laumond. “Dynamic path modification for car-like nonholonomic mobile robots.” In: *IEEE International Conference on Robotics and Automation*. 1997, pp. 2920–2925 (cited on p. 3).

[151] O. Khatib. “The Potential Field Approach and Operational Space Formulation in Robot Control.” In: *Workshop on Applications of Adaptive Systems Theory*. 1985, pp. 208–214 (cited on p. 60).

[152] O. Khatib. “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots.” In: *International Journal of Robotics Research* 5.1 (1986), pp. 90–98 (cited on p. 11).

[153] J. M. Kilner, Y. Paulignan, and S. J. Blakemore. “An Interference Effect of Observed Biological Movement on Action.” In: *Current Biology* 13.6 (2003), pp. 522–525 (cited on pp. 2, 10).

[154] J. Kilner, A. F. d. C. Hamilton, and S.-J. Blakemore. “Interference effect of observed human movement on action is due to velocity profile of biological motion.” In: *Social Neuroscience* 2.3-4 (2007), pp. 158–66 (cited on p. 10).

[155] J.-O. Kim and P. K. Khosla. “Real-time obstacle avoidance using harmonic potential functions.” In: *IEEE Transactions on Robotics* 8.3 (1992), pp. 338–349 (cited on p. 63).

[156] S. Kim, C. Kim, B. You, and S Oh. “Stable Whole-body Motion Generation for Humanoid robots to Imitate Human Motions.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009 (cited on p. 10).

[157] L. Kobbelt, T. Bareuther, and H.-P. Seidel. “Multiresolution shape deformations for meshes with dynamic vertex connectivity.” In: *Computer Graphics Forum* 19.3 (2000) (cited on p. 59).

[158] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. “Interactive multi-resolution modeling on arbitrary meshes.” In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 1998, pp. 105–114 (cited on p. 28).

[159] L. Kobbelt, J. Vorsatz, and H.-P. Seidel. “Multiresolution hierarchies on unstructured triangle meshes.” In: *Computational Geometry* 14.1-3 (1999), pp. 5–24 (cited on p. 59).

[160] J. Koenemann, F. Burget, and M. Bennewitz. “Real-time imitation of human whole-body motions by humanoids.” In: *IEEE International Conference on Robotics and Automation*. 2014, pp. 2806–2812 (cited on p. 10).

[161] A. Kolobov, Mausam, and D. S. Weld. “A Theory of Goal-Oriented MDPs with Dead Ends.” In: *Conference on Uncertainty in Artificial Intelligence*. 2012, pp. 438–447 (cited on p. 117).

[162] A. Kolobov, Mausam, and D. S. Weld. “A Theory of Goal-Oriented MDPs with Dead Ends.” In: 2012 (cited on p. 118).

[163] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell. “Upper-body Kinesthetic Teaching of a Free-standing Humanoid Robot.” In: *IEEE International Conference on Robotics and Automation*. 2011, pp. 3970–3975 (cited on p. 2).

[164] D. Kortenkamp and R. Simmons. “Robotic Systems Architectures and Programming.” In: *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008, pp. 187–206 (cited on p. 6).

[165] V. Krüger, D. Kragic, and C. Geib. “The meaning of action: A review on action recognition and mapping.” In: *Advanced Robotics* 21.13 (2007), pp. 1473–1501 (cited on p. 3).

[166] B. Krüger, J. Baumann, M. Abdallah, and A. Weber. “A Study On Perceptual Similarity of Human Motions.” In: *Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*. 2011 (cited on p. 4).

[167] B. Kühnlenz, S. Sosnowski, M. BuSS, D. Wollherr, K. Kühnlenz, and M. Buss. “Increasing Helpfulness towards a Robot by Emotional Adaption to the User.” In: *International Journal of Social Robotics* (2013), pp. 1–20 (cited on p. 114).

[168] D. Kulić and E. A. Croft. “Safe planning for human-robot interaction.” In: *Journal of Robotic Systems* 22.7 (2005), pp. 383–396 (cited on p. 114).

[169] D. Kulic and Y. Nakamura. “Comparative study of representations for segmentation of whole body human motion data.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 4300–4305 (cited on p. 62).

[170] D. Kulic, W. Takano, and Y. Nakamura. “Incremental Learning, Clustering and Hierarchy Formation of Whole Body Motion Patterns using Adaptive Hidden Markov Chains.” In: *International Journal of Robotics Research* 27.7 (2008), pp. 761–784 (cited on p. 62).

[171] D. Kulic, W. Takano, and Y. Nakamura. “Online Segmentation and Clustering From Continuous Observation of Whole Body Motions.” In: *IEEE Transactions on Robotics* 25.5 (2009), pp. 1158–1166 (cited on p. 63).

[172] H. Kunori, D. Lee, and Y. Nakamura. "Associating and reshaping of whole body motions for object manipulation." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 5240–5247 (cited on p. 3).

[173] A. Kupferberg, S. Glasauer, M. Huber, M. Rickert, A. Knoll, and T. Brandt. "Biological movement increases acceptance of humanoid robots as human partners in motor interaction." In: *AI & Society* 26.4 (2011), pp. 339–345 (cited on pp. 4, 10).

[174] H. Kwakernaak. *Linear Optimal Control Systems*. Ed. by R. Sivan. John Wiley & Sons, Inc., 1972 (cited on p. 63).

[175] F. Lamiraux, D. Bonnafous, and O. Lefebvre. "Reactive path deformation for nonholonomic mobile robots." In: *IEEE Transactions on Robotics* 20.6 (2004), pp. 967–977 (cited on pp. 11, 13).

[176] J.-F. Landry, J.-P. Dussault, and P. Mahey. "Billiards: an optimization challenge." In: *International C* Conference on Computer Science and Software Engineering*. 2011, pp. 129–132 (cited on p. 115).

[177] J.-F. Landry and J.-P. Dussault. "AI Optimization of a Billiard Player." In: *Journal of Intelligent and Robotic Systems* 50 (2007), pp. 399–417 (cited on pp. 115, 120, 124, 136).

[178] B. Lau, C. Sprunk, and W. Burgard. "Kinodynamic motion planning for mobile robots using splines." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 2427–2433 (cited on p. 11).

[179] S. M. Lavalle. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. No. 98-11, 1998 (cited on p. 11).

[180] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006 (cited on p. 3).

[181] S. M. LaValle and J. J. Kuffner. "Randomized Kinodynamic Planning." In: *International Journal of Robotics Research* 20.5 (2001), pp. 378–400 (cited on p. 11).

[182] D. A. Lawrence, E. W. Frew, and W. J. Pisano. "Lyapunov Vector Fields for Autonomous Unmanned Aircraft Flight Control." In: *Journal of Guidance, Control, and Dynamics* 31.5 (2008), pp. 1220–1229 (cited on p. 63).

[183] W. Leckie and M. Greenspan. "An event-based pool physics simulator." In: *International Conference on Advances in Computer Games*. 2005, pp. 247–262 (cited on p. 115).

[184] D. Lee and C. Ott. "Incremental kinesthetic teaching of motion primitives using the motion refinement tube." In: *Autonomous Robots* 31.2-3 (2011), pp. 115–131 (cited on p. 2).

[185] D. Lee, M. Morf, and B. Friedlander. "Recursive least squares ladder estimation algorithms." In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 29.3 (1981), pp. 627–641 (cited on p. 80).

[186] J.-G. Lee, J. Han, X. Li, and H. Gonzalez. "TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering." In: *Proceedings of the VLDB Endowment* 1 (1 2008), pp. 1081–1094. ISSN: 2150-8097 (cited on p. 63).

[187] J. Lee, O. Kwon, L. Zhang, and S.-E. Yoon. “SR-RRT: Selective retraction-based RRT planner.” In: *IEEE International Conference on Robotics and Automation*. 2012, pp. 2543–2550 (cited on p. 11).

[188] K. Leibrandt, T. Lorenz, T. Nierhoff, and S. Hirche. “Modelling Human Gameplay at Pool and Countering it with an Anthropomorphic Robot.” In: *International Conference on Social Robotics*. 2013, pp. 30–39 (cited on pp. 8, 136).

[189] S. Levine and V. Koltun. “Continuous Inverse Optimal Control with Locally Optimal Examples.” In: *International Conference on Machine Learning*. 2012 (cited on p. 11).

[190] B. Levy. “Laplace-Beltrami Eigenfunctions Towards an Algorithm That "Understands" Geometry.” In: *Shape Modeling International*. 2006, pp. 13– (cited on p. 13).

[191] 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 (cited on pp. 63, 73).

[192] Z. M. Lin, J.-S. Yang, and C. Y. Yang. “Grey decision-making for a billiard robot.” In: *IEEE International Conference on Systems, Man and Cybernetics*. 2004, pp. 5350–5355 (cited on pp. 115, 120, 136).

[193] S. Lindemann and S. L. Valle. “Smoothly Blending Vector Fields for Global Robot Navigation.” In: *IEEE Conference on Decision and Control*. 2005, pp. 3353–3559 (cited on p. 63).

[194] Y. Lipman, O. Sorkine, M. Alexa, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. “Laplacian Framework for Interactive Mesh Editing.” In: *International Journal of Shape Modeling* 11.1 (2005), pp. 43–62 (cited on p. 17).

[195] Y. Lipman, O. Sorkine, D. C.-O. D. Levin, C. Rössl, and H.-P. Seidel. “Differential Coordinates for Interactive Mesh Editing.” In: *Shape Modeling International*. 2004, pp. 181–190 (cited on p. 20).

[196] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. “Linear rotation-invariant coordinates for meshes.” In: *ACM Transactions on Graphics* 24.3 (2005), pp. 479–487 (cited on p. 20).

[197] U. Luxburg. “A tutorial on spectral clustering.” In: *Statistics and Computing* 17.4 (2007), pp. 395–416 (cited on p. 13).

[198] S. Martini, M. Egerstedt, and A. Bicchi. “Controllability decompositions of networked systems through quotient graphs.” In: *IEEE Conference on Decision and Control*. 2008, pp. 5244–5249 (cited on p. 59).

[199] D. Matsui, T. Minato, K. F. MacDorman, and H. Ishiguro. “Generating natural motion in an android by mapping human motion.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 3301–3308 (cited on p. 2).

[200] J. R. Medina, M. Lawitzky, A. Mörtl, D. Lee, and S. Hirche. “An experience-driven robotic assistant acquiring human knowledge to improve haptic cooperation.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 2416–2422 (cited on pp. 63, 114).

[201] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos. "A multiresolution symbolic representation of time series." In: *IEEE Internation Conference on Data Engineering*. 2005, pp. 668–679 (cited on p. 63).

[202] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds." In: *Visualization and Mathematics* (2002), pp. 35–57 (cited on pp. 13, 18).

[203] M. B. Milam. "Real-time optimal trajectory generation for constrained dynamical systems." PhD thesis. California Institute of Technology, 2003 (cited on p. 63).

[204] K. Miura, H. Furukawa, and M. Shoji. "Similarity of human motion: congruity between perception and data." In: *Systems, Man and Cybernetics*. 2006, pp. 1184–1189 (cited on p. 2).

[205] Y. Mohan and S. Ponnambalam. "An extensive review of research in swarm robotics." In: *Nature Biologically Inspired Computing*. 2009, pp. 140–145 (cited on p. 10).

[206] K. Mombaur, A.-H. Olivier, and A. Crétual. "Forward and Inverse Optimal Control of Bipedal Running." In: *Modeling, Simulation and Optimization*. Vol. 18. Springer Berlin Heidelberg, 2013, pp. 165–179 (cited on p. 11).

[207] K. Mombaur, A. Truong, and J.-P. Laumond. "From human to humanoid locomotion - an inverse optimal control approach." In: *Autonomous Robots* 28.3 (2010), pp. 369–383 (cited on p. 11).

[208] A. D. Moore. *Mechanics of Billiards, and Analysis of Willie Hoppe's Stroke*. 1942. URL: http://www.sfbilliards.com/AD_MOORE.PDF (cited on p. 148).

[209] B. Morris and M. Trivedi. "Learning trajectory patterns by clustering: Experimental studies and comparative evaluation." In: *Conference on Computer Vision and Pattern Recognition*. 2009, pp. 312–319 (cited on p. 63).

[210] J.-M. Morvan and B. Thibert. "On the approximation of a smooth surface with a triangulated mesh." In: *Computational Geometry* 23.3 (2002), pp. 337–352 (cited on p. 18).

[211] T. Moulard, E. Yoshida, and S. Nakaoka. "Optimization-based Motion Retargeting Integrating Spatial and Dynamic Constraints for Humanoid." In: *International Symposium on Robotics*. 2013, FA2–4 (cited on p. 3).

[212] A. Mörtl, T. Lorenz, B. N. S. Vlaskamp, A. Gusrialdi, A. Schubö; and S. Hirche. "Modeling inter-human movement coordination: synchronization governs joint task dynamics." In: *Biological Cybernetics* 106.4-5 (2012), pp. 241–259 (cited on p. 10).

[213] M. Muhlig, M. Gienger, J. Steil, and C. Goerick. "Automatic selection of task spaces for imitation learning." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 4996–5002 (cited on p. 62).

[214] N. Mukosaka, I. Tanev, and K. Shimohara. "Performance of Incremental Genetic Programming on Adaptability of Snake-like Robot." In: *Procedia Computer Science* 24.0 (2013), pp. 152 –157 (cited on p. 2).

[215] H. Müller and D. Sternad. “Decomposition of variability in the execution of goal-oriented tasks: three components of skill improvement.” In: *Journal of Experimental Psychology: Human Perception and Performance* 30.1 (2004), pp. 212–233 (cited on p. 2).

[216] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, and A. M. Erkmen. “Search and Rescue Robotics.” In: *Springer Handbook of Robotics*. Springer Verlag, 2008, pp. 1151–1173 (cited on p. 4).

[217] Y. Nakamura. *Advanced robotics - redundancy and optimization*. 1991 (cited on pp. 62, 66).

[218] S. Nakaoka and T. Komura. “Interaction mesh based motion adaptation for biped humanoid robots.” In: *IEEE-RAS International Conference on Humanoid Robots*. 2012, pp. 625–631 (cited on p. 111).

[219] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. “Laplacian mesh optimization.” In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 2006, pp. 381–389 (cited on p. 13).

[220] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Sch ölkopf. “Learning Inverse Dynamics: A Comparison.” In: *European Symposium on Artificial Neural Networks*. 2008 (cited on p. 63).

[221] K. Nickel and R. Stiefelhagen. “Visual recognition of pointing gestures for human-robot interaction.” In: *Image and Vision Computing* 25.12 (2007), pp. 1875–1884 (cited on p. 114).

[222] T. Nierhoff, K. Heunisch, and S. Hirche. “Strategic play for a pool-playing robot.” In: *IEEE Workshop on Advanced Robotics and its Social Impacts*. 2012, pp. 72–78 (cited on pp. 8, 134, 136).

[223] T. Nierhoff and S. Hirche. “Fast Trajectory Replanning Using Laplacian Mesh Optimization.” In: *IEEE International Conference on Control, Automation, Robotics and Vision*. 2012, pp. 154–159 (cited on pp. 7, 18, 59).

[224] T. Nierhoff and S. Hirche. “Trajectory Classification in n Dimensions using Subspace Projection.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 1318–1323 (cited on pp. 8, 111).

[225] T. Nierhoff, S. Hirche, and Y. Nakamura. “Multiresolution Laplacian Trajectory Replanning.” In: *Proceedings of the Annual Conference of RSJ*. 2013 (cited on pp. 7, 59).

[226] T. Nierhoff, S. Hirche, and Y. Nakamura. “Variable Positional Constraints for Laplacian Trajectory Editing.” In: *DGR-Tage*. 2013 (cited on pp. 7, 59).

[227] T. Nierhoff, S. Hirche, and Y. Nakamura. “Laplacian Trajectory Vector Fields for Robotic Movement Imitation and Adaption.” In: *CISM-IFToMM SYMPOSIUM on Theory and Practice of Robots and Manipulators*. 2014 (cited on pp. 8, 111).

[228] T. Nierhoff, S. Hirche, and Y. Nakamura. “Sampling-based trajectory imitation in constrained environments using Laplacian-RRT.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 3012–3018 (cited on pp. 7, 59).

[229] T. Nierhoff, S. Hirche, W. Takano, and Y. Nakamura. “Full Body Motion Adaption based on Task-Space Distance Meshes.” In: *IEEE International Conference on Robotics and Automation*. 2014, pp. 1865 –1870 (cited on pp. 8, 111).

[230] T. Nierhoff, O. Kourakos, and S. Hirche. “Playing pool with a Dual-Armed Robot.” In: *IEEE International Conference on Robotics and Automation*. 2010, pp. 3445 – 3446 (cited on pp. 8, 122, 134, 136).

[231] S. S. Obhi and N Sebanz. “Moving together: Towards understanding the mechanisms of joint action.” In: *Experimental Brain Research* 211 (2011), pp. 329–336 (cited on p. 114).

[232] D. T. T. Office. *DARPA Robotics Challenge*. DARPA-BAA-12-39. 2015 (cited on p. 4).

[233] Y. Ogura, H. Aikawa, K. Shimomura, H. Kondo, A. Morishima, H. ok Lim, and A. Takanishi. “Development of a New Humanoid Robot WABIAN-2.” In: *IEEE International Conference on Robotics and Automation*. 2006, pp. 76–81 (cited on p. 4).

[234] A. Oikonomopoulos, I. Patras, M. Pantic, and N. Paragios. “Trajectory-based representation of human actions.” In: *International Joint Conferences on Artificial Intelligence*. 2007, pp. 133–154 (cited on p. 63).

[235] M. Okada, K. Tatani, and Y. Nakamura. “Polynomial design of the nonlinear dynamics for the brain-like information processing of whole body motion.” In: *IEEE International Conference on Robotics and Automation*. 2002, pp. 1410–1415 (cited on pp. 63, 90).

[236] V. Okunev, T. Nierhoff, and S. Hirche. “Human-preference-based Control Design: Adaptive Robot Admittance Control for Physical Human-Robot Interaction.” In: *IEEE International Workshop on Robots and Human Interactive Communications*. 2012, pp. 443 –448 (cited on p. 2).

[237] E. Oztop, D. W. Franklin, T. Chaminade, and G. Cheng. “Human-Humanoid Interaction: is a Humanoid Robot Perceived as a Human?” In: *International Journal of Humanoid Robotics* 2.4 (2005), pp. 537–559 (cited on p. 10).

[238] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal. “Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields.” In: *IEEE-RAS International Conference on Humanoid Robots*. 2008, pp. 91–98 (cited on p. 11).

[239] F. C. Park and B. Ravani. “Smooth Invariant Interpolation of Rotations.” In: *ACM Transactions on Graphic* 16.3 (1997), pp. 277–295 (cited on p. 139).

[240] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. “Learning and generalization of motor skills by learning from demonstration.” In: *IEEE International Conference on Robotics and Automation*. 2009, pp. 1293–1298 (cited on p. 2).

[241] S. N. Patek, W. L. Korff, and R. L. Caldwell. “Deadly strike mechanism of a mantis shrimp.” In: *Nature* 428 (2004), pp. 819–820 (cited on p. 1).

[242] A. Pekarovskiy and M. Buss. “Optimal control goal manifolds for planar nonprehensile throwing.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 4518–4524 (cited on p. 51).

[243] A. Perez, R. Platt, G. Konidaris, L. P. Kaelbling, and T. Lozano-Pérez. "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics." In: *IEEE International Conference on Robotics and Automation*. 2012, pp. 2537–2542 (cited on p. 11).

[244] Q.-C. Pham and Y. Nakamura. "Affine trajectory deformation for redundant manipulators." In: *Robotics: Science and Systems*. 2012 (cited on p. 11).

[245] Q.-C. Pham and Y. Nakamura. "A New Trajectory Deformation Algorithm Based on Affine Transformations." In: *International Joint Conference on Artificial Intelligence*. 2013 (cited on pp. 11, 30, 63).

[246] U. Pinkall, S. D. Juni, and K. Polthier. "Computing Discrete Minimal Surfaces and Their Conjugates." In: *Experimental Mathematics* 2 (1993), pp. 15–36 (cited on p. 18).

[247] A. P. Pobegailo. "Spherical splines and orientation interpolation." In: *The Visual Computer* 11 (1 1994), pp. 63–68 (cited on p. 139).

[248] K. Polthier. "Computational aspects of discrete minimal surfaces." In: *Global Theory of Minimal Surfaces*. 2005 (cited on p. 13).

[249] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. "ROS: an open-source Robot Operating System." In: *ICRA Workshop on Open Source Software*. 2009 (cited on p. 133).

[250] S. Quinlan and O. Khatib. "Elastic Bands: Connecting Path Planning and Control." In: *IEEE International Conference on Robotics and Automation*. 1993, pp. 802–807 (cited on pp. 11, 13).

[251] M. Reuter, S. Biasotti, D. Giorgi, G. Patanè, and M. Spagnuolo. "Discrete Laplace-Beltrami operators for shape analysis and segmentation." In: *Computational Geometry* 33.3 (2009), pp. 381–390 (cited on p. 13).

[252] C. W. Reynolds. "Flocks, Herds and Schools: A Distributed Behavioral Model." In: *ACM Special Interest Group on Graphics and Interactive Techniques* 21.4 (1987), pp. 25–34 (cited on p. 1).

[253] E. Rimon and D. E. Koditchek. "Exact robot navigation using artificial potential functions." In: *IEEE Transactions on Robotics and Automation* 5 (1992), pp. 501–518 (cited on p. 11).

[254] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras. "Learning collaborative impedance-based robot behaviors." In: *AAAI Conference on Artificial Intelligence*. 2013, pp. 1422–1428 (cited on p. 2).

[255] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010 (cited on p. 124).

[256] L. Saab, O. Ramos, F. Keith, N. Mansard, P. Soueres, and J. Fourquet. "Dynamic Whole-Body Motion Generation Under Rigid Contacts and Other Unilateral Constraints." In: *IEEE Transactions on Robotics* 29.2 (2013), pp. 346–362 (cited on p. 63).

[257] A. Salazar and A. Sanchez-Lavega. "Motion of a ball on a rough horizontal surface after being struck by a tapering rod." In: *European Journal of Physics* 11 (1990), pp. 228–232 (cited on p. 148).

[258] P. Sardain and G. Bessonnet. “Forces acting on a biped robot. Center of pressure-zero moment point.” In: *IEEE Transactions on Systems, Man, and Cybernetics* 34.5 (2004), pp. 630–637 (cited on p. 67).

[259] A. H. Sayed. “Lattice Filter Algorithms.” In: *Adaptive Filters*. John Wiley & Sons, Inc., 2008, pp. 669–675 (cited on p. 80).

[260] S. Schaal. “Dynamic Movement Primitives - A Framework for Motor Control in Humans and Humanoid Robots.” In: *International Symposium on Adaptive Motion of Animals and Machines*. 2003 (cited on pp. 11, 90).

[261] S. Schaal. “Dynamic Movement Primitives - A Framework for Motor Control in Humans and Humanoid Robotics.” In: *Adaptive Motion of Animals and Machines* (2006). Ed. by H Kimura, K Tsuchiya, A Ishiguro, and H. Witte, pp. 261–280 (cited on pp. 11, 13).

[262] J. P. Scholz and G. Schöner. “The uncontrolled manifold concept: identifying control variables for a functional task.” In: *Experimental Brain Research* 126.3 (1999), pp. 289–306 (cited on p. 10).

[263] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. “Motion planning with sequential convex optimization and convex collision checking.” In: *International Journal of Robotics Research* 33.9 (2014), pp. 1251–1270 (cited on pp. 11–13).

[264] P. Scokaert and J. Rawlings. “Constrained linear quadratic regulation.” In: *IEEE Transactions on Automatic Control* 43.8 (1998), pp. 1163–1169 (cited on p. 63).

[265] N. Sebanz, H. Bekkering, and G. Knoblich. “Joint action: bodies and minds moving together.” In: *Trends in Cognitive Sciences* 10.2 (2006), pp. 70–76 (cited on p. 114).

[266] N. Sebanz and G. Knoblich. “Prediction in Joint Action: What, When, and Where.” In: *Topics in Cognitive Science* 1.2 (2009), pp. 353–367 (cited on p. 114).

[267] I. R. Shaiju A. J. andPetersen. “Formulas for Discrete Time LQR, LQG, LEQG and Minimax LQG Optimal Control Problems.” In: *International Federation of Automatic Control World Congress*. 2008 (cited on p. 63).

[268] J. Q. Shi, R. Murray-Smith, M. Titterington, and J. Q. Shi. “Bayesian Regression and Classification Using Mixtures of Gaussian Processes.” In: *International Journal of Adaptive Control and Signal Processing* (2003), pp. 149–161 (cited on p. 12).

[269] X. Shi, K. Zhou, Y. Tong, M. Desbrun, H. Bao, and B. Guo. “Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics.” In: *ACM Transactions on Graphics* 26.3 (2007) (cited on p. 62).

[270] A. Shkolnik. “Sample-Based Motion Planning in High-Dimensional and Differentially-Constrained Systems.” PhD thesis. MIT, 2010 (cited on p. 43).

[271] A. C. Shkolnik and R. Tedrake. “Path planning in 1000+ dimensions using a task-space Voronoi bias.” In: *IEEE International Conference on Robotics and Automation*. 2009, pp. 2061–2067 (cited on p. 43).

[272] K. Shoemake. "Animating Rotation with Quaternion Curves." In: *ACM International Conference on Computer Graphics and Interactive Techniques*. Vol. 19. 3. 1985, pp. 245–254 (cited on p. 30).

[273] J. Shoshani. "Proboscidea (Elephants)." In: *Encyclopedia of Life Sciences*. John Wiley & Sons, Ltd, 2001 (cited on p. 1).

[274] E. C. e Silva, F. Costa, E. Bicho, and W. Erlhagen. "Nonlinear optimization for human-like movements of a high degree of freedom robotics arm-hand system." In: *International Conference on Computational Science and Applications*. Vol. 3. 2011, pp. 327–342 (cited on p. 3).

[275] W. D. Smart. "Making reinforcement learning work on real robots." PhD thesis. Brown University, 2002 (cited on p. 62).

[276] M. Smith. "PickPocket: A computer billiards shark." In: *Artificial Intelligence* 171 (2007), pp. 1069–1091 (cited on pp. 115, 120, 136).

[277] E. D. Sontag. "Nonlinear regulation, the piecewise linear approach." In: *Princeton Conference on Information Sciences and Systems*. 1980 (cited on p. 63).

[278] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. "Laplacian Surface Editing." In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 2004, pp. 175–184 (cited on p. 17).

[279] O. Sorkine. "Laplacian Mesh Processing." In: *EUROGRAPHICS STAR - State of the Art Report* (2005) (cited on p. 13).

[280] O. Sorkine and M. Alexa. "As-rigid-as-possible surface modeling." In: *Eurographics Symposium on Geometry Processing*. 2007, pp. 109–116 (cited on pp. 21, 22, 29, 59).

[281] O. Sorkine and D. Cohen-Or. "Least-Squares Meshes." In: *Shape Modeling International*. 2004, pp. 191–199 (cited on p. 13).

[282] O. Sorkine, D. Cohen-Or, and S. Toledo. "High-pass quantization for mesh encoding." In: *Eurographics Symposium on Geometry Processing*. 2003, pp. 42–51 (cited on p. 13).

[283] W. A. Sparrow. "The efficiency of skilled performance." In: *Journal of Motor Behavior* 15 (1983), pp. 237–261 (cited on p. 2).

[284] M. Sreenivasa. "Modeling of human movement for the generation of humanoid robot motion." PhD thesis. Institute National Polytechnique de Toulouse, France, 2012 (cited on p. 10).

[285] R. S. Strichartz. "Analysis of the Laplacian on the Complete Riemannian Manifold." In: *Journal of Functional Analysis* 52.1 (1983), pp. 48–79 (cited on p. 13).

[286] S. Stronge, Victoria, and A. Museum. *Tipu's Tigers*. V & A Publishing, 2009 (cited on p. 1).

[287] O. von Stryk and R. Bulirsch. "Direct and Indirect Methods for Trajectory Optimization." In: *Annals of Operations Research* 37.1-4 (1992), pp. 357–373 (cited on p. 3).

[288] C. W. Studer. "Augmented time-stepping integration of non-smooth dynamical systems." PhD thesis. ETH Zürich, 2008 (cited on p. 152).

[289] T. Sugihara, Y. Nakamura, and H. Inoue. "Realtime Humanoid Motion Generation through ZMP Manipulation Based on Inverted Pendulum Control." In: *IEEE International Conference on Robotics and Automation*. 2002, pp. 1404–1409 (cited on p. 87).

[290] W. Suleiman, E. Yoshida, F. Kanehiro, J.-P. Laumond, and A. Monin. "On human motion imitation by humanoid robot." In: *IEEE International Conference on Robotics and Automation*. 2008, pp. 2697–2704 (cited on p. 2).

[291] W. Takano, H. Imagawa, and Y. Nakamura. "Prediction of human behaviors in the future through symbolic inference." In: *IEEE International Conference on Robotics and Automation*. 2011, pp. 1970–1975 (cited on p. 63).

[292] W. Takano, K. Yamane, T. Sugihara, K. Yamamoto, and Y. Nakamura. "Primitive Communication based on Motion Recognition and Generation with Hierarchical Mimesis Model." In: *IEEE International Conference on Robotics and Automation*. 2006, pp. 3602–3609 (cited on p. 85).

[293] G. Taubin. "A signal processing approach to fair surface design." In: *ACM Special Interest Group on Graphics and Interactive Techniques*. 1995, pp. 351–358 (cited on pp. 18, 19, 29).

[294] G. Taubin. "Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation." In: *International Conference on Computer Vision*. 1995, pp. 902–907 (cited on p. 13).

[295] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts. "LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification." In: *International Journal of Robotics Research* 29.8 (2010), pp. 1038–1052 (cited on p. 63).

[296] *The PR2 Plays Pool*. Willow Garage. 2010. URL: `http://www.willowgarage.com/blog/2010/06/15/pr2-plays-pool` (cited on p. 115).

[297] M. Thielscher. "Reasoning About Actions: Steady Versus Stabilizing State Constraints." In: *Artificial Intelligence* 104.1-2 (1998), pp. 339–355 (cited on p. 114).

[298] S. Thompson and S. Kagami. "Continuous Curvature Trajectory Generation with Obstacle Avoidance for Car-Like Robots." In: *Computational Intelligence for Modelling, Control and Automation*. Vol. 1. 2005, pp. 863–870 (cited on p. 3).

[299] S. Thrun. "Toward a framework for human-robot interaction." In: *Human-Computer Interaction* 19.1 (2004), pp. 9–24 (cited on p. 2).

[300] A. N. Tikhonov. "Solution of incorrectly formulated problems and the regularization method." In: *Doklady Akademii Nauk SSSR* 151 (1963), pp. 501–504 (cited on p. 111).

[301] L. H. Ting and J. L. McKay. "Neuromechanics of muscle synergies for posture and movement." In: *Current opinion in neurobiology* 17.6 (2007), pp. 622–8 (cited on p. 10).

[302] M. de la Torre Juarez. "The effect of impulsive forces on a system with friction: the example of the billiard game." In: *European Journal of Physics* 15 (1994), pp. 184–190 (cited on p. 148).

[303] M. Toussaint, N. Plath, T. Lang, and N. Jetchev. "Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference." In: *IEEE International Conference on Robotics and Automation*. 2010, pp. 385–391 (cited on p. 3).

[304] S. Umeyama. "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (1991), pp. 376–380 (cited on p. 20).

[305] M. M. Veloso and P. Stone. "Video: RoboCup robot soccer history 1997 - 2011." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5452–5453 (cited on p. 115).

[306] M. Vlachos, G. Kollios, and D. Gunopulos. "Discovering Similar Multidimensional Trajectories." In: *IEEE Internation Conference on Data Engineering*. 2002, pp. 673–684 (cited on pp. 63, 110).

[307] O. Von Stryk and M. Schlemmer. "Optimal control of the industrial robot Manutec r3." In: *Computational Optimal Control* 115 (1994), pp. 367–382 (cited on p. 51).

[308] R. E. Wallace and M. C. Schroeder. "Analysis of billiard ball collisions in two dimensions." In: *American Journal of Physics* 56, no.9 (1988), pp. 815–819 (cited on p. 148).

[309] R. E. Wallace and M. C. Schroeder. "Analysis of billiard ball collisions in two dimensions." In: *American Journal of Physics* 56, no.9 (1988), pp. 815–819 (cited on p. 152).

[310] Z. Wang, M. Deisenroth, H. B. Amor, D. Vogt, B. Scholkopf, and J. Peters. "Probabilistic Modeling of Human Movements for Intention Inference." In: *Robotics: Science and Systems*. 2012 (cited on p. 114).

[311] M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun. "Discrete laplace operators: no free lunch." In: *Eurographics Symposium on Geometry Processing*. 2007, pp. 33–37 (cited on p. 29).

[312] D. J. Webb and J. van den Berg. "Kinodynamic RRT*: Optimal Motion Planning for Systems with Linear Differential Constraints." In: *Computing Research Repository* abs/1205.5088 (2012) (cited on p. 59).

[313] J. Witters and D. Duymelinck. "Rolling and sliding resistive forces on balls moving on a flat surface." In: *American Journal of Physics* 54, no.1 (1986), pp. 80–83 (cited on p. 148).

[314] D. M. Wolpert and Z. Ghahramani. "Computational principles of movement neuroscience." In: *Nature Neuroscience* 3 (2000), pp. 1212–1217 (cited on p. 10).

[315] G. Wood. *Living Dolls: A Magical History of the Quest for Mechanical Life*. Faber & Faber, 2003 (cited on p. 1).

[316] S. Wu and Y. Li. "On Signature Invariants for Effective Motion Trajectory Recognition." In: *International Journal of Robotics Research* 27 (8 2008), pp. 895–917 (cited on p. 63).

[317] S. Wu, Y. Li, and J. Zhang. "A hierarchical motion trajectory signature descriptor." In: *IEEE International Conference on Robotics and Automation*. 2008, pp. 3070–3075 (cited on p. 63).

[318] G. Xu. “Convergent Discrete Laplace-Beltrami Operators over Triangular Surfaces.” In: *Geometric Modeling and Processing*. 2004, pp. 195–204 (cited on p. 13).

[319] K. Yamane and Y. Nakamura. “Dynamics Filter - concept and implementation of online motion Generator for human figures.” In: *IEEE Transactions on Robotics and Automation* 19 (2003), pp. 421–432 (cited on p. 3).

[320] G. Ye and R. Alterovitz. “Demonstration-Guided Motion Planning.” In: *International Society of Root Research* (2011) (cited on p. 11).

[321] E. Yoshida and F. Kanehiro. “Reactive robot motion using path replanning and deformation.” In: *IEEE International Conference on Robotics and Automation*. 2011, pp. 5456–5462 (cited on p. 3).

[322] T. Yoshikawa. “Manipulability and redundancy control of robotic mechanisms.” In: *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. Vol. 2. 1985, pp. 1004–1009 (cited on p. 66).

[323] H. Zhang. “Discrete Combinatorial Laplacian Operators for Digital Geometry Processing.” In: *SIAM Conference on Geometric Design*. 2004, pp. 575–592 (cited on p. 18).

[324] H. Zhang, O. van Kaick, and R. Dyer. “Spectral Mesh Processing.” In: *Computer Graphics Forum* 29.6 (2010), pp. 1865–1894 (cited on pp. 13, 18).

[325] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. “Large mesh deformation using the volumetric graph Laplacian.” In: *ACM Transactions on Graphics* 24.3 (2005), pp. 496–503 (cited on p. 13).

[326] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. A. D. Bagnell, and S. Srinivasa. “CHOMP: Covariant Hamiltonian Optimization for Motion Planning.” In: *International Journal of Robotics Research* 32 (2013), pp. 1164–1193 (cited on pp. 11, 13).