TECHNISCHE UNIVERSITÄT WIEN

ACIN
AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

Path Planning and Path Following Control for Mechanical Systems

DIPLOMARBEIT

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Univ.-Prof. Dr. techn. Andreas Kugi
Dr. techn. Patrik Zips
Dipl.-Ing. Florian Schausberger

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by

Michael Schwegel
Matriculation number 0926940
Untere Augartenstraße 21 Top 19
1020 Vienna
Austria

Vienna, August 2016

Preamble

This diploma thesis with the title *Path Planning and Path Following Control for Mechanical Systems* was conducted at the Automation and Control Institute of Vienna University of Technology.

What started out as a method for path-following of a multi rotor helicopter lead to a general approach applicable to a variety of mechanical systems. Based on recent literature the proposed controller was intended to solve some inherent issues of available approaches.

I would like to express my gratitude to my supervisors Dr. techn. Patrik Zips and Dipl.-Ing. Florian Schausberger for their help with words and deeds and all other members of the institute who made this work possible. I especially thank Univ.-Prof. Dr. techn. Andreas Kugi for his commitment and enquiring interest in my work, which sparked new enthusiasm every time. Furthermore, I want to thank Dipl.-Ing Bernhard Bischof for many insightful discussions and his contributions to this work. Dipl.-Ing Florian Rötzer, Dipl.-Ing Mario Fohler and Max Meraner BSc. stand symbolically for endless and inspiring discussions amongst colleagues and friends which I deeply cherish.

Finally, I want to thank my parents Cornelia and Peter for their ongoing support and encouraging words. It is a tremendous gift to have a family which one can always rely on. Similarly, my greatest appreciation goes to my brother Florian and sister Hanna who motivate me to challenge myself and improve at every step of the way.

Vienna, August 2016

Abstract

The goal of this thesis is to provide systematic means to plan and follow a path for mechanical systems. Therefore, this thesis is divided into two main parts, one for each of these problems.

Firstly, the path planning problem is introduced and then solved by using the so-called Rapidly Exploring Random Tree algorithm. To improve the so-found paths, a modification is introduced which renders the solutions optimal with respect to a specific criterion. To avoid high computation times, the algorithm was simplified by consistent assumptions on the mechanical system.

Secondly, a path following controller is derived to follow the calculated path in presence of disturbances and to account for modelling errors. Using this controller, it is possible to control the velocity along the path with one control law and a second control law is defined to account for deviations from the path. To the best of the authors knowledge, this controller, unlike for any other approach, can be specified in fixed coordinates. This particularly simple approach is applicable to a wide variety of systems, in particular mechanical systems.

In simulation the proposed concepts were tested for a number of mechanical systems. To underline its practical feasibility the path following approach was validated using a parallel kinematic robot.

Kurzzusammenfassung

Ziel dieser Arbeit ist die methodische Behandlung einer Pfadplanung und die anschließende Pfadfolgeregelung für mechanische Systeme. Der vorliegende Text ist in zwei Teile gegliedert, die jeweils einem dieser Probleme gewidmet sind.

Im ersten Teil wird mittels der so genannten Rapidly Exploring Random Trees (schnell erkundende zufällige Bäume) ein allgemeines Pfadplanungsproblem gelöst. Dieses besteht aus der Forderung einen Pfad zwischen zwei definierten Punkten eines Raumes zu finden, der durch Hindernisse beschränkt ist. Mittels einer Erweiterung des Algorithmus können optimale Pfade, im Sinne eines spezifizierten Kriteriums, gefunden werden. Durch geeignete Annahmen lässt sich dieses Problem auf eine weite Klasse mechanischer Systeme anwenden.

Der so gewonnene Pfad soll weiters durch ein mechanisches System gefolgt werden. Zu diesem Zweck wurde ein neuer Ansatz für eine Pfadfolgeregelung entwickelt. Hierbei ist nicht nur die Verfolgung des Pfades, sondern auch eine unabhängige Spezifikation der Reaktion auf Abweichungen möglich.

Alle beschriebenen Methoden werden anhand geeigneter Simulationen analysiert und ausgewertet. Diesbezüglich wurden mechanische Systeme mit unterschiedlichen Charakteristika herangezogen, um die gewählten Ansätze zu illustrieren. Die Validierung der beschriebenen Methoden mithilfe eines drei-achsigen Parallelroboters untermauert deren praktische Anwendbarkeit.

Contents

1 Introduction

Industrial manufacturing is recently undergoing a trend towards flexible mass production. This means that a factory has to be able to reconfigure its production with less effort, even for large scale processes. Furthermore, the number of customisable products and the use of decentralised systems in factories are significantly increasing. To cope with these developments flexible production processes are required. Especially for automatic systems, pre-planned movements have to be replaced by online algorithms. For example a welding robot needs to be able to adjust to different shaped products and thus the welding path has to be adopted accordingly. In this context, many tasks include the movement of mechanical systems, like robots, along a path. Practically, any such movement can be divided into a planning and a control problem.

The former is called path planning and describes the search for a collision free path in a given environment. The main challenge in path planning is to avoid obstacles and to achieve a suitable solution path within reasonable time bounds. An overview of planning algorithms can be found in, e. g., [1, 2]. In this thesis, an algorithm is presented and analysed which searches for the shortest path between two points. Here, particular emphasis is placed on the fact that the presented algorithm improves its solution gradually and converges to the shortest path. Thus, depending on the available time, the planning algorithm can be cancelled anytime while still providing a feasible path. Furthermore, the underlying obstacle avoidance is implemented as simple boolean collision detection, which is easily adapted to a variety of problems.

To control a mechanical system along a path many algorithms have been proposed, see, e. g., [3]. However, many well known control approaches cannot ensure that the geometric reference path is thoroughly respected. Intuitively, this can lead to the use of shortcuts which are a significant problem for various applications, e. g., machining, welding, etc. This thesis focuses on a path following algorithm, which guarantees that the tool stays on the geometrical reference path. The main contribution is a novel path following control law which uses fixed coordinates to facilitate the practical implementation.

This thesis is divided into two main chapters that deal with the path planning and path following problem, respectively. In Chapter 2, a path planning algorithm is introduced, which searches for a collision free path between two points. In the following sections, this algorithm is extended to search for the shortest path. A mechanical system can be controlled along a path by the novel path following controller, which is presented in Chapter 3. There, after the basic methods are introduced, the admissible paths are specified. Geometric observations then lead to a problem-specific view on the system by which the proposed controller is introduced. The path following controller and the combination of path planning and following are validated in extensive numerical simulations and a practical experiment in Chapter 4.

2 Path Planning

The term path planning is used in a variety of disciplines and has several meanings depending on the context. In the following text it is understood as finding feasible paths connecting two states of a dynamical system, while avoiding obstacles.

Depending on the problem to be solved, path planning deals with *continuous*-time dynamical systems or *discrete*-time systems. The first of which can be described by nonlinear differential equations, the latter is governed by discrete motion laws which can be formulated as algebraic equations. The algorithms discussed in the scope of this chapter can only be applied to discrete-time systems. To apply them to continuous systems a discretisation scheme can be applied.

This chapter is structured into three main parts: First, Sections 2.1 and 2.2 give an overview of the field and establish the basic notions, respectively. Second, the problem to be solved is stated in Section 2.3. Third, an algorithm to solve this problem is introduced in Section 2.4 and further investigated in Section 2.5.

2.1 Literature Review

This section gives a brief overview of path planning. The research in path planning started in the late 1960's alongside the development of the first computer-controlled robots. Beginning in the 1980's however, when robots started becoming a vital part of industrial processes, the field has received more and more attention. Robotics is still a growing field: space exploration, undersea and hazardous environment work or crisis intervention are only some examples, with increasing use of robotics. Path planning is necessary for each robotic motion and therefore it is a fundamental task in robotics. An extensive overview of the field can be found in [2, 4] and more recently [1].

Early approaches were prevalently based on a variety of heuristics and approximate methods. Such methods are for example grid based search methods or the concatenation of curve segments, so-called clothoids. Algorithms with strong guarantees on successful termination have been studied extensively in the 1990's, see, e. g., [5–7]. Furthermore, the problems were generalised to systems of higher complexity, such as cars, which cannot turn without driving. However, these methods tend to be very complex or even computationally intractable. Two ideas which led to the development of practical planning methods were spatial discretisation and the use of potential fields, see [8] and [9], respectively. The former is a discrete approximation of the problem to improve computational tractability. With this simplification the strict conditions on viable solutions were relaxed to resolution dependent conditions. On the other hand, optimal path planning using potential fields cannot, without modifications, guarantee to satisfy constraints. Thus, it is not possible to ensure a collision free path, cf., [10]. Despite their drawbacks, methods following these

two ideas demonstrate remarkable performance, solving complex tasks within reasonable time, see, e. g., [11].

Broader applicability of these methods was achieved by using randomisation and navigation functions, see [12] and [13], respectively. By introducing random exploration of the space the relevant path planning problems could be solved more efficiently. However, due to their statistical basis only weaker properties can be established. Such methods, like the *Probabilistic Road Map* or the hereafter treated *Rapidly Exploring Random Tree*, are often based on graph structures. By statistically building a graph of feasible path segments the path planning problem reduces to a graph search. These algorithms have shown good performance for a variety of complex problems, see, e. g., [14, 15]. These approaches were modified to address a plethora of specific problems. Among these are: local optimal methods, path planning on manifolds, nonholonomic planning, time critical planning and others.

A detailed historical perspective on path planning can be found in [1, 4].

2.2 Basic Notions

The following section introduces basic notions of path planning and establishes a basis to categorise and evaluate an algorithm. In path planning for robotic systems, the whole configuration of the robot has to be considered. For example, the configuration of articulated robots can be described by the angles and positions of its joints. For a simple car model its position and orientation have to be considered. These variables form the configuration space of a robotic system, which completely characterise the system. Hence, any motion of a robot can be considered a path in the configuration space. Such a path has to avoid collisions and satisfy constraints, thus has to lie in the so-called free configuration space, a subset of the configuration space. For consistency with the following chapters, the state and output spaces are used in favour of the configuration space. The state space is a set of variables fully describing the dynamical state of the system, including its velocities. Linking this idea to the configuration space of a robot the output space is introduced. The output space is a collection of state variables or a function thereof. Intuitively, the output function is a mapping which describes some quantity of interest in terms of the state. In the scope of this work, the output space is limited to position states and thus can be regarded as a subspace of the configuration space. The trajectory of a system can thus be regarded as a path in the output space. In this sense, path planning is considering the connection of two output states by a feasible path.

Early contributions in the field dealt mostly with what is known as *feasible* path planning. This term describes the problem of finding any feasible and thus collision free path without considering additional specifications like the length of the path. Clearly, it is desirable for any planning algorithm to minimise some measure of cost like the required actuation energy or the path length. These *optimal* planning algorithms are mostly based on necessary optimality conditions, e. g., the Jacobi-Hamilton-Bellmann equations, the calculus of variations or dynamic programming. These methods can be further categorised into local and global methods, depending on the scope of the underlying optimality condition. Local methods require an initial candidate solution and search for an optimum

only in a local neighbourhood of this initial guess. Thus, a local optimisation may not be able to find the global optimal solution depending on the initialisation. On the other hand, local methods are typically much less computationally expensive.

Deterministic algorithms are, in the simplest case, based on the idea that each state can be visited in finite time. Therefore, such algorithms are said to be *complete*. A complete path planner terminates in finite time, returning a valid solution if one exists, or failure otherwise. Because they search over the entire free space such methods do not scale well to higher dimensional problems.

To overcome this limitation, sampling-based methods were introduced. However, the strict notion of completeness cannot be achieved by sampling-based methods. An important factor for such methods is their sampling strategy. A sampling approach is called *dense*, if the samples are drawn arbitrarily close to any state in a finite state space. Many sampling-based path planning algorithms are based on uniform random sampling, which is dense with probability one. Methods based on dense sampling are *probabilistically complete*, if the probability to find a solution, provided that one exists, converges to one as the number of samples increases. Otherwise, if no solution exists the algorithm may run in an infinite loop. Their most important property, i. e. the rate of convergence, is usually very difficult to establish. Thus, one has to rely on simulations of benchmarking problems.

Furthermore, iterative path planning can be divided into *depth-* and *breadth*-first search. Algorithms are categorised as breadth-first when exhaustively searching for short local paths, while a depth-first search explores long deep paths before searching locally.

Randomised algorithms are in particular of practical interest when used with complicated systems and environments.

2.3 Problem Statement

The subject of this chapter is the solution of a discrete optimal path planning problem for holonomic systems. This is achieved by first considering the corresponding feasible planning problem in Section 2.4 and a subsequent extension to the optimal problem in Section 2.4.1.

The feasible path planning problem has the following properties, cf. [1],

- a nonempty set of feasible output states $\mathcal{Y}_{free} \neq \{\}$ which can be finite or countably infinite,
- an initial state $\mathbf{x}_0$ such that $\mathbf{y}_0 = \mathbf{h}(\mathbf{x}_0) \in \mathcal{Y}_{free}$ and
- a goal state $\mathbf{y}_g \in \mathcal{Y}_{free}$.

The notation of a state as $\mathbf{x}$ is not to be confused with an output state. A point in the output space, here denoted by $\mathbf{y}$, describes a characteristic position of the system only. Velocities or angle states are not treated here, as discussed in more detail in Chapter 5.

The problem is to find a feasible path $\boldsymbol{\sigma}(\lambda)$ such that $\mathbf{y}_0 = \boldsymbol{\sigma}(\lambda_0)$ and $\mathbf{y}_g = \boldsymbol{\sigma}(\lambda_g)$. This parametrised form of a curve $\boldsymbol{\sigma}(\lambda)$ is found as interpolation of states connecting the initial

with the goal state in Section 2.4.2. The set of these states is called the solution of the Rapidly Exploring Random Tree (RRT) and needs to be entirely contained in the free output space. Note that this problem formulation does not provide any means to follow the path.

The problem of finding such points can be easily expressed as graph search. Therefore, each output point is considered a node in the graph. A directed edge between two vertices is called input $\mathbf{u}$. An input between states is feasible only for states with a connection lying entirely in the free space. Furthermore, such an input has to transfer the output between these points according to the system dynamics. It is required here that there always exists such an action, if the straight connection between the points lies in the free space. The initial and goal states are marked by special vertices. The search for a solution path is considered as the problem of building a graph with feasible nodes. This graph is searched for a viable solution path to the problem. The RRT algorithm is a method to build such a graph. It is used to solve a single planning problem and is thus a single-query algorithm. A multi-query algorithm similar to the RRT is called Probabilistic Road Map. This variation generates a graph with edges from each node to each other node, which can be used repeatedly for a given space.

2.4 Rapidly Exploring Random Trees

In the following section, a basic algorithm for the discrete feasible path planning problem is introduced. The Rapidly Exploring Random Tree (RRT) was originally published in [16]. It is an iterative algorithm based on the idea to gradually increase the density of its search tree. New nodes are added to the tree stochastically by connecting random vertices to the graph. These randomly sampled nodes are connected to the tree by edges to the closest node in the tree according to some distance metric. Using the explored states, the graph is used to find a feasible path from the initial state to the goal state.

The RRT can be applied to a broad spectrum of problems. However, finding a suitable distance metric between states for a nonlinear system is not straightforward. Furthermore, the states can consist of elements with different physical meaning like angles and positions. In such cases, the planning problem can be simplified to a position output space without considering the system dynamics. Because of this simplification, the algorithm does not yield the required inputs to the system to follow the resulting path. This drawback can be overcome by employing a path following controller. This approach is pursued in the following and therefore the planning problem is restricted to a search on the output space $\mathcal{Y} \subset \mathbb{R}^p$. With this strategy, uniform sampling covers the finite space densely. Thus, the RRT on a subset of $\mathbb{R}^p$ is probabilistically complete.

A RRT is generated by connecting a random sample $\mathbf{y}_s$ to the closest node stored in the tree $\mathbf{y}_c$. Starting with a graph containing only the initial (root) node, it is connected to the first random sample such that $\mathbf{y}_0 = \mathbf{y}_c$, forming an edge. If the tree contains nodes other than the initial node, new connections are formed to the closest node in the tree $\mathbf{y}_c$. The general case of adding a sample $\mathbf{y}_s$ to the tree is illustrated in Figure 2.1. Here, the new sample $\mathbf{y}_s$ is connected to the nearest neighbouring sample contained in the tree $\mathbf{y}_c$.

To avoid straight connections through the whole space, the step size $\mathbf{y}_c - \mathbf{y}_s = \mathbf{u}$ is

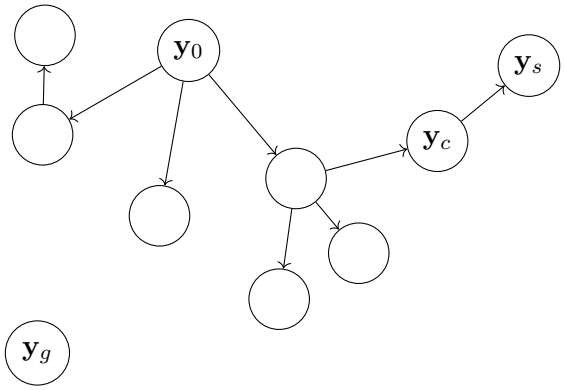

Figure 2.1: A RRT with initial node $\mathbf{y}_0$, where a random sample state $\mathbf{y}_s$ is added to its closest neighbouring node $\mathbf{y}_c$. The goal node $\mathbf{y}_g$ has not been connected to the tree, thus there is no solution path connecting the start and goal nodes.

limited to a maximum length $\|\mathbf{u}\| \leq u_{max}$. If the distance of the sample to its closest neighbour is larger, the sample is modified. This modified sample $\mathbf{y}_s^*$ is constructed to be u_{max} in distance away from the closest node in the direction of the original sample

$$\mathbf{y}_s^* = \mathbf{y}_c + u_{max} \frac{(\mathbf{y}_s - \mathbf{y}_c)}{\|\mathbf{y}_s - \mathbf{y}_c\|} \ .$$

Note that in general such a linear interpolation is only possible without considering the system dynamics. It is worth noting that by the use of this interpolation the space is not sampled densely. However, a modification is only applied when the tree does not contain a node closer than u_{max} to a new sample. Thus, this interpolation strategy has no effect as the space is successively covered with nodes. For a sparse tree, however, the interpolation generates samples closer to the tree which are more likely to have collision free connections. These observations justify the use of the modification to decrease the number of required samples while retaining dense sampling at a high number of nodes.

Note that in the general case one has to find a feasible input between the sampled state and its closest point in the tree. Therefore, the system dynamics

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

and the output equation

$$\mathbf{y}_{k+1} = \mathbf{h}(\mathbf{x}_{k+1})$$

are used to find an input to the system, which connects a sampled node to its closest neighbour by solving the optimisation problem

$$\mathbf{u}_k = \arg \min_{\mathbf{u}_k \in \mathcal{U}} \|\mathbf{y}_{k+1} - \mathbf{y}_s\|$$

for the corresponding input. Here, the current state $\mathbf{x}_k = \mathbf{x}_c$ and the state $\mathbf{x}_{k+1}$ is calculated such that the corresponding output $\mathbf{y}_{k+1}$ is as close as possible to the sample

and thus

$$\mathbf{y}_{k+1} = \mathbf{h}\left(\mathbf{f}(\mathbf{x}_c, \mathbf{u}_k)\right) \approx \mathbf{y}_s$$

holds. The proof of global optimality for this scheme is similar to the proof of optimality for value iteration, cf. [1].

Since the output space $\mathcal{Y}$ in general contains obstacles, the samples are drawn only from the obstacle-free subset of the state space $\mathcal{Y}_{free}$. A uniformly distributed sample of $\mathcal{Y}_{free}$ can be drawn by rejection-sampling. Therefore, a sample is drawn uniformly from the output space $\mathcal{Y}$ and checked for a collision with the obstacles. If the sample lies in $\mathcal{Y}_{free}$ it can be used, otherwise another sample is drawn. Using this method the states added to the tree are guaranteed to be in the free space. However, one has to make sure that the edges connecting those states are collision free as well. The collision detection is discussed in further detail in Section 2.4.3.

The algorithm presented so far exhibits a characteristic similar to a breadth-first search. However, it does not yield a feasible path planner, since the expansion sample can never be the goal state. This can be explained by the fact that the probability that the drawn sample is the goal state is zero. Inevitably, the tree cannot expand a node which is exactly the goal state. To circumvent this fact, the algorithm has to be biased towards the goal state. This is done by explicitly choosing the expansion sample to be the goal state with a nonzero probability. The bias leads to a directed expansion of the tree between the initial and goal state. Thus, the search paradigm of an RRT can be shifted towards depth-first search by increasing the weight of the bias.

After the goal node is expanded as a node in the tree, the algorithm is terminated. The edges of this node are a unique series of states from the goal node to the initial node, thus solving the problem. A flow chart of the RRT algorithm is depicted in Figure 2.2.

It is worth noting that statistically the tree is expanded in bigger steps whenever the tree is sparse. This is intuitively true since there are less nodes to expand to, thus the average edge length is large. On the other hand, for a dense tree the probability that a node in the tree lies close to a random sample is high. Hence, in a dense tree, short connections are more probable to be explored. This is illustrated using two RRTs with a different number of nodes in Figure 2.3. Here, the starting point is marked by a circle, the goal by a cross and the blue line marks the final path. The obstacles are depicted in light grey. Note that the sparse tree has nodes with long edges far into the search space whereas the increasing number of samples decreases the average connection length. However, this favourable characteristic that the algorithm increases its level of detail with increased samples cannot be used in an RRT. After finding any solution the algorithm is terminated returning the first valid solution. Therefore, information gained by further exploration of the free space is not used in this basic form of the algorithm. In Section 2.4.1, a modification of the RRT algorithm is introduced, which uses the available information in the tree to improve the solution with respect to a cost function.

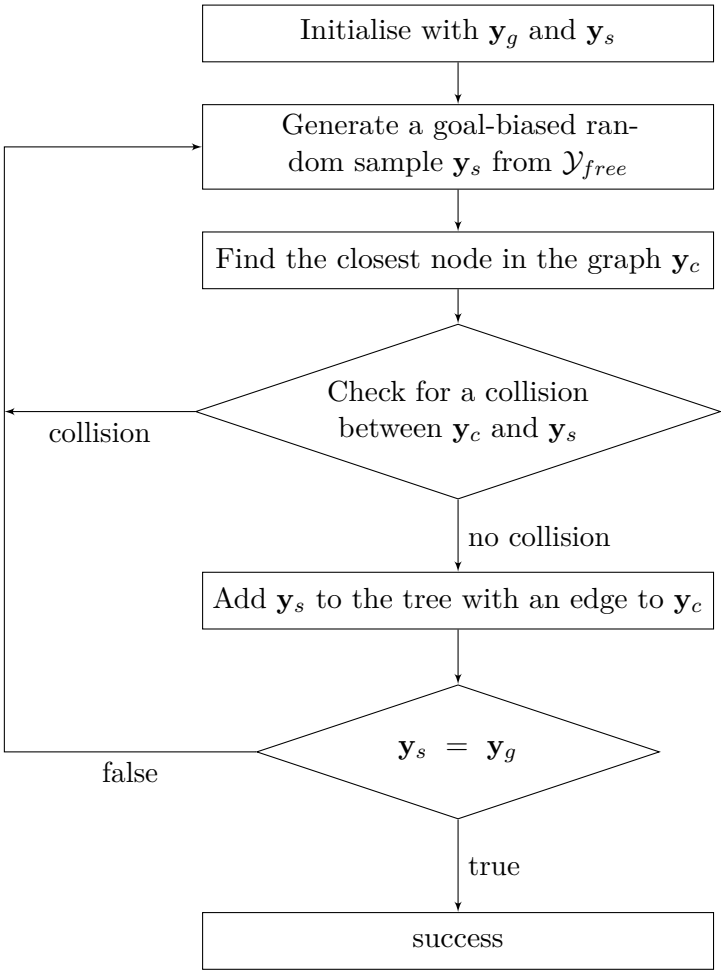

Figure 2.2: Flow diagram of the RRT algorithm. In addition to the termination condition $\mathbf{y}_s = \mathbf{y}_g$ a maximum sample count can be used to terminate the algorithm.

2.4.1 Optimal Rapidly Exploring Random Trees

In the following, the RRT algorithm is modified to find a path which minimises a cost function. The solution path thus should fulfil

$$\tilde{\mathbf{y}} = \arg\min_{\tilde{\mathbf{y}}} J(\tilde{\mathbf{y}}) \ , \tag{2.1}$$

where $\tilde{\mathbf{y}} = \{\mathbf{y}_0, \ldots, \mathbf{y}_n\}$ are the $n+1$ nodes in the solution path and the positive definite function

$$J : \mathcal{Y}^{n+1} \to \mathbb{R}^+$$

is called the cost function. To calculate the minimum of the cost function iteratively, it is assumed to have an additive structure. Since the planning is carried out in the output

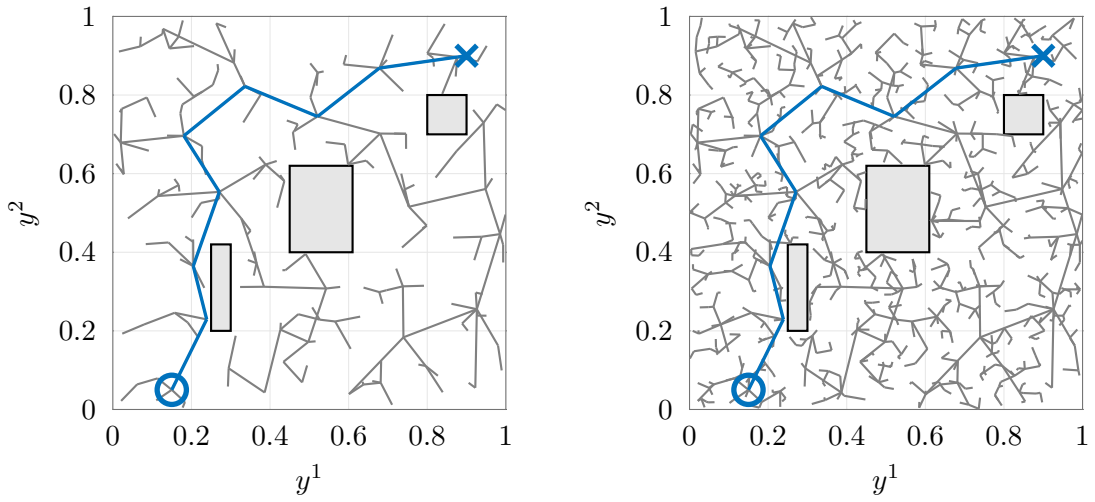

Figure 2.3: Two RRTs for a two-dimensional problem with 3 rectangular obstacles. Each edge is represented by a line. The sparse tree on the left contains 200 nodes, whereas the dense tree on the right was expanded with 1000 nodes. The RRT solution does not change after a valid solution is found, therefore the solution path (blue) is equivalent.

space the length of the path is used as cost function

$$J(\tilde{\mathbf{y}}) = \sum_{i=0}^{n} \|\mathbf{y}_{i+1} - \mathbf{y}_i\|$$

for a path along $n+1$ nodes. The cost function gives the distance from a node to the initial node, the so-called root node. Therefore, global optimality can be enforced in each individual step, rather than being an implicit condition on the whole path, cf. value iteration algorithms.

To solve this optimal path planning problem, a modification of the RRT algorithm of Section 2.4 originally proposed in [15] is presented. This optimal stochastic planning algorithm is based on the idea to use the information of each node to gradually decrease the cost of the edges in the tree. Every node, except for the root node, is connected to exactly one node by an edge. This node is called the parent node. The optimal RRT algorithm differs from the RRT in two steps:

Firstly, a newly sampled node $\mathbf{y}_s$ is not connected to the closest node in the tree but to the node with minimum costs $\mathbf{y}_a$. Therefore, the tree is searched for other nodes in a neighbouring set of nodes $\mathcal{V}$ around $\mathbf{y}_s$ to find the node $\mathbf{y}_a$ which yields the lowest cost function. Note that although $\mathbf{y}_c$ is the closest node to the new sample $\mathbf{y}_s$ there can be a node with lower costs, because the cost function depends on all connected nodes up until the root node. Figure 2.4 depicts an example for a new node $\mathbf{y}_s$. Here, connecting $\mathbf{y}_s$ to the closest node $\mathbf{y}_c$ yields overall costs of $J = 7 + 1 = 8$ as shown on the left-hand side. On the right-hand side the new node is connected to the optimal node $\mathbf{y}_a$ which yields

the lower cost function $J = 3 + 3 = 6$. Note that the new node is only connected to the optimum node $\mathbf{y}_a$ and not to the closest node $\mathbf{y}_c$ to retain the tree structure of the graph.

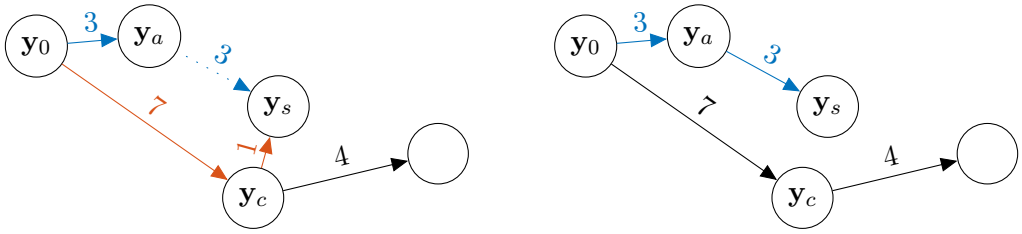

Figure 2.4: The optimal connection of a new sample. Edges are depicted with their corresponding contribution to the cost function. After the basic RRT algorithm established a connection with the closest node $\mathbf{y}_c$ (red path) the original connection is replaced by the cheapest path to the root node (blue path).

Secondly, after sampling a new node the edges of the existing tree are replaced with edges of lower costs from the new node. Thus, not only paths which end in the new sample $\mathbf{y}_s$ are optimised but also existing edges are optimised via the new sample. An example for this second step is depicted in Figure 2.5. Here, the connection of the existing node $\mathbf{y}_b$

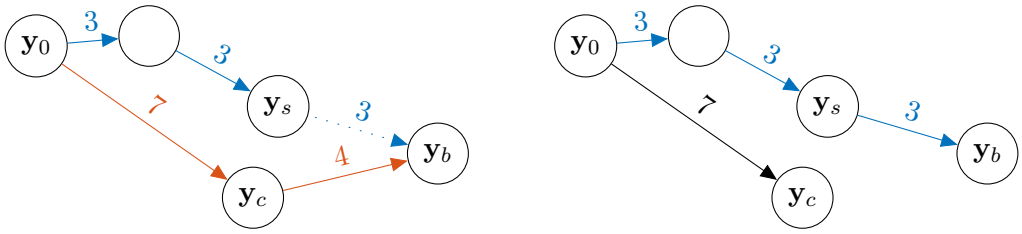

Figure 2.5: If there is a cheaper connection via the new sampled node $\mathbf{y}_s$ (blue path) its edge replaces the original edge (red path). Edges are depicted with their corresponding contribution to the cost function.

via $\mathbf{y}_c$ with total costs of $J = 7 + 4 = 11$ is replaced by a new connection via the new node which yields $J = 3 + 3 + 3 = 9$.

A flow diagram for the optimal RRT algorithm is depicted in Figure 2.6. It is clearly illustrated that the optimal RRT differs from the regular RRT only in the two steps where nodes with optimal edges replace the trivial connections of the RRT. Since this algorithm converges to the optimal solution and the solution is improved statistically with each node. Thus, it has to be terminated after reaching a sufficiently large number of samples N_{max} or after a defined computation time.

The aforementioned modifications involve checking every sample in the neighbourhood $\mathcal{V}$ around $\mathbf{y}_s$ twice. Thus, it shall be as small as possible. However, the size of this set is crucial for the optimality of the algorithm. To implicitly define this set a radius $r \in \mathbb{R}$ around the new sample is introduced such that the neighbourhood is defined by

$$\mathcal{V} := \{\mathbf{y} \in \mathcal{G} \mid \|\mathbf{y} - \mathbf{y}_s\| \leq r\}$$

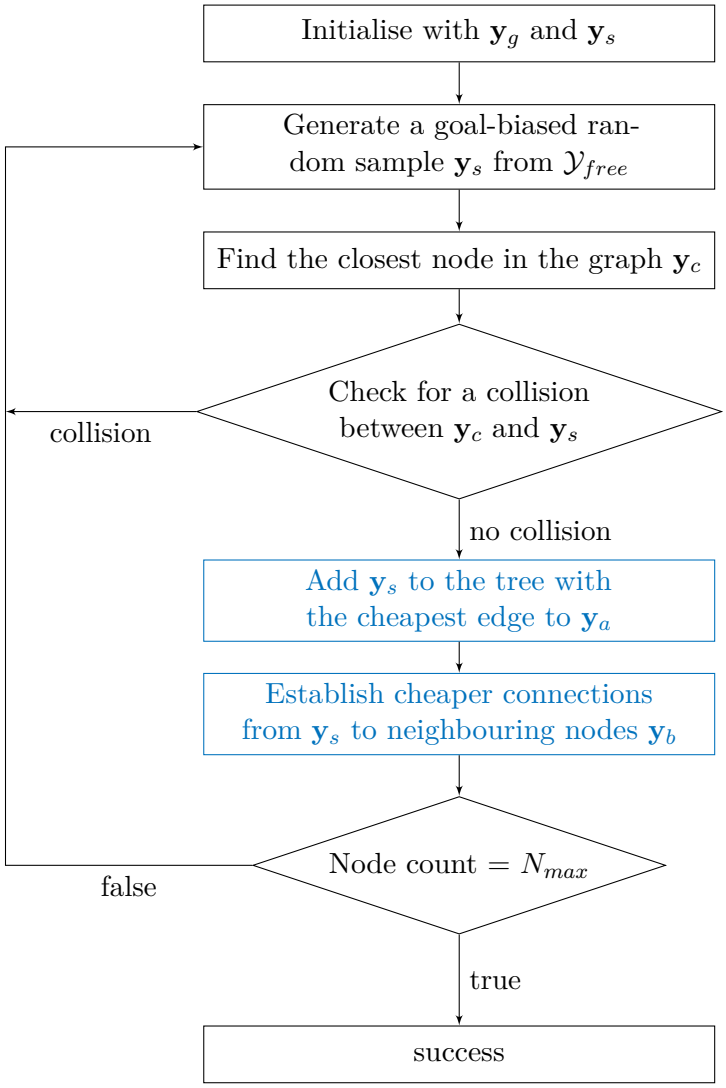

Figure 2.6: Flow diagram of the optimal RRT algorithm. The algorithm is terminated after the graph contains a specified number of nodes.

with $\mathcal{G}$ the graph of the RRT, which consists of all its nodes. In [15] it is proven that there exists a minimal radius r such that the algorithm is asymptotically optimal. However, this number depends on the cardinality of the graph and is not explicitly computable. It is trivial to see that the maximum step size u_{max} is an upper bound for this radius. Thus, the maximum step size is a good estimate of the radius r. Since it is out of the scope of this work to prove the optimality of the given method, the interested reader is referred to [15]. Intuitively, the resulting path is globally asymptotically optimal by the argument that each single step is optimised. Using the additive structure this compares directly to results on optimality of dynamic programming and value iteration, see [17].

An example of the optimal RRT using a simple two-dimensional path planning problem is depicted in Figure 2.7. Here, the starting point is marked by a circle and the goal by a

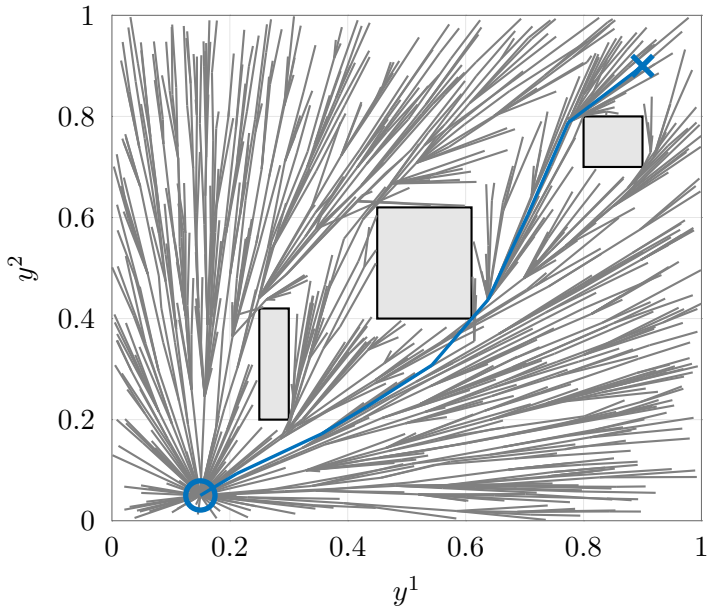

Figure 2.7: Optimal RRT for the two-dimensional planning problem. Edges are represented by lines and the final path is coloured in blue. The tree was expanded with 1000 nodes using the same random samples used in Figure 2.3.

cross. Despite being sampled from the same random samples as the trees in Figure 2.3, the modifications lead to a significant change. Note that there are locally optimal solutions to this problem above and below each obstacle. Intuitively, it is a big change to switch between those solutions across the obstacle. Moreover, the regular RRT with the same random samples is closest to the local optimum above the obstacles. However, the optimal RRT still converges to the global optimum in between the obstacles.

2.4.2 Cubic Spline Interpolation

The RRT algorithm, after successful termination, returns a set of states between the initial and goal state $\tilde{\mathbf{y}} = \{\mathbf{y}_0, \ldots, \mathbf{y}_g\}$. These discrete points are connected by collision free linear connections. However, this piecewise linear path is not continuously differentiable. To find a path representation which is suitable for the path following algorithm developed in Section 3.3.2 the solution graph of the RRT is interpolated. Because of its wide popularity in industry and favourable properties, cubic splines are used here to interpolate the states in the graph. The main benefit of a cubic spline interpolation is the good trade-off between smoothness and simplicity of calculation. Polynomial interpolation algorithms tend to oscillations at the outermost points to be interpolated. This effect is a result of the high order polynomials tending to infinity. By interpolating separate low order polynomials

these oscillations are avoided. The main principle of a cubic spline interpolation is to interpolate a cubic polynomial between each pair of points.

In this section, an interpolation of the $n+1$ points $\mathbf{y}_k$, $k = 0, \ldots, n$ obtained from an RRT is presented. The path is defined by n polynomials of third order along the intervals $i = 0, \ldots, n-1$

$$\boldsymbol{\sigma}_i(\lambda) = \mathbf{a}_i \, \lambda^3 + \mathbf{b}_i \, \lambda^2 + \mathbf{c}_i \, \lambda + \mathbf{d}_i \ , \tag{2.2}$$

with coefficients $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d} \in \mathbb{R}^p$. Each curve segment $\boldsymbol{\sigma}_i$ is defined between $\mathbf{y}_i$ and $\mathbf{y}_{i+1}$. For the splines to pass through the given points, there has to be a parameter value λ such that

$$\boldsymbol{\sigma}_k(\lambda_k) = \mathbf{y}_k \ , \tag{2.3}$$

holds for some $\lambda = \lambda_k$ for each point. These are $n+1$ conditions for the $n+1$ points. The coefficients can be found by using this interpolation condition and by enforcing continuity across adjacent curve segments. The curve segments have to be continuous at their shared points which yields

$$\boldsymbol{\sigma}_{j-1}(\lambda_j) = \boldsymbol{\sigma}_j(\lambda_j) = \mathbf{y}_j \ . \tag{2.4}$$

Here, $j = 1, \ldots, n-1$, since all points, except for the first and last one are covered by two adjacent curve segments. These are algebraic equations for all n segments $\boldsymbol{\sigma}_i$. Furthermore, the tangent needs to be continuous at the interpolation points and this holds true if and only if

$$\boldsymbol{\sigma}'_{j-1}(\lambda_j) = \boldsymbol{\sigma}'_j(\lambda_j) \ , \tag{2.5}$$

where $(\cdot)'$ denotes the derivation with respect to the path parameter λ. Similarly, the condition for a continuous curvature is

$$\boldsymbol{\sigma}''_{j-1}(\lambda_j) = \boldsymbol{\sigma}''_j(\lambda_j) \ . \tag{2.6}$$

Equations (2.4) to (2.6) require that the segmented path yields continuous derivatives up to the second order at the connection points between two segments. Since the segments are polynomials, of order three, the interpolated path is twofold continuously differentiable, i. e. $\boldsymbol{\sigma} \in \mathcal{C}^2$. Subsequently, it is assumed that a set of coordinates for the p-dimensional output space is chosen. This is always possible since cartesian position coordinates are assumed. Because of the independence of the coordinates the calculation of the polynomials (2.2) can be performed separately for each component. Subsequently the interpolation is shown for the y^1-y^2 plane and the superscript 1 is omitted to improve readability. A component of (2.2) reads

$$\sigma_i(\lambda) = a_i \, \lambda^3 + b_i \, \lambda^2 + c_i \, \lambda + d_i \ .$$

Substitution of this equation into (2.3) yields $d_i = y_i^2$ for $\lambda = y - y_i$. By expansion of (2.4) to (2.6) with (2.2), the conditions on interpolation and continuity yield a set of linear conditions on b_m, $m = 1, \ldots, n-1$.

$$\mathbf{Y} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} r_1 - b_0(y_1 - y_0) \\ r_2 \\ \vdots \\ r_{n-2} \\ r_{n-1} - b_n(y_n - y_{n-1}) \end{bmatrix} \tag{2.7}$$

where the known right-hand side term r_m is given by

$$r_m = 3\left(\frac{d_{m+1} - d_m}{y_{m+1} - y_m} - \frac{d_m - d_{m-1}}{y_m - y_{m-1}}\right) .$$

The coefficient matrix

$$\mathbf{Y} = \begin{bmatrix} 2(y_2 - y_0) & y_2 - y_1 & 0 & \dots & 0 \\ y_2 - y_1 & 2(y_3 - y_1) & y_3 - y_2 & & \vdots \\ 0 & & \ddots & & 0 \\ \vdots & & y_{n-2} - y_{n-3} & 2(y_{n-1} - y_{n-3}) & y_{n-1} - y_{n-2} \\ 0 & \dots & 0 & y_{n-1} - y_{n-2} & 2(y_n - y_{n-2}) \end{bmatrix}$$

depends only on the interpolation points. The terms b_0 and b_n are not determined by either of the stated equations. They are equal to half the curvature of the path at the outermost points, y_0 and y_{n+1}, respectively. These values were chosen to be 0 to avoid excessive curvature in the start and end segment. The linear system of equations (2.7) contains only known quantities, except for the coefficients b_i. Therefore, it can be solved for these coefficients by any solver for systems of linear equations. The *Thomas*-Algorithm was used to solve this problem efficiently by exploiting the tridiagonal structure, see [18]. This method can be used for diagonal dominant tridiagonal systems. By summation of the columns of $\mathbf{Y}$ it is easy to see that the problem is diagonal dominant and can be solved with this approach. After determination of the coefficients d_i and b_i it follows from (2.3), (2.5) and (2.6) that

$$a_i = \frac{b_{i+1} - b_i}{3(y_{i+1} - y_i)} \tag{2.8}$$

and

$$c_i = \frac{d_{i+1} - d_i}{y_{i+1} - y_i} - \frac{1}{3}(b_{i+1} + 2b_i)(y_{i+1} - y_i) , \tag{2.9}$$

which can be solved by back substitution for all remaining coefficients $i = 0, \dots, n-1$. These calculations have to be done for each coordinate in order to complete the map $\boldsymbol{\sigma}(\lambda)$. The choice $\lambda = y - y_i$ can be changed into any other strictly monotonously increasing function in y.

2.4.3 Collision Detection

To plan a collision free path, the RRT algorithm requires a boolean collision detection method. Subsequently, it is assumed that the system can be sufficiently represented by a point in the output space. If this is not the case, the obstacles can be increased in size to keep a minimal distance between the system and any obstacle. Additionally, virtual obstacles can be used to prevent collisions of unmodelled parts. There are two types of collision detection used in the context of this work, namely a point-box and a line-box collision detection. These types represent the collision of a single state with an obstacle and the collision of an edge with an obstacle, respectively. Obstacles of different shapes

can be approximated as a combination of rectangular obstacles. The RRT algorithm only requires a simple boolean collision detection, because each extension step can be checked separately. The presented collision check methods are based on a per-axis decomposition of the problem. Thus, problem is reduced to simple one dimensional intersection problems along some axis.

Point-Box Collision

To sample the free space using rejection-sampling, a sampled random state has to be checked if it lies in the free space. Therefore, the sampled point in the output space is checked against all the obstacles. This can be easily achieved by approximating the obstacles by one or more rectangular boxes. The point collides with an obstacle if each coordinate of the point lies between the maximum and minimum coordinates of any such box. Figure 2.8 illustrates this situation. Here, the y^1 coordinate of the point is between the maximum and minimum y^1 coordinate of the rectangle. Likewise, the y^2 coordinates of the obstacle are above and below the point. The point lies in the free space if either of these conditions is not met. This method is easily generalised to higher dimensions by

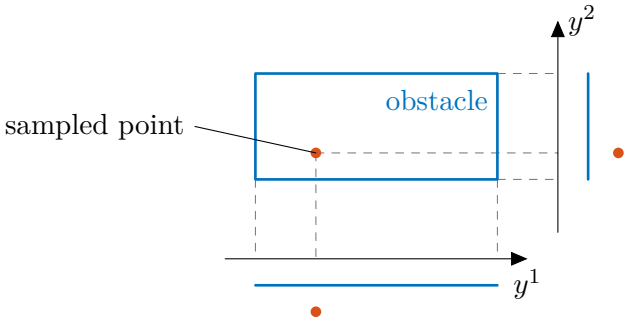

Figure 2.8: A rectangular obstacle enclosing a sampled point.

including a check for each axis. Thus, a point lies within a d-dimensional box, if and only if the i-coordinate of the point y_p^i satisfies

$$y_{b,1}^i < y_p^i < y_{b,2}^i$$

for all axes y^i with $i = 1, \ldots, d$ and $y_{b,1}^i, y_{b,2}^i$ the maximum and minimum coordinate value of the box, respectively. This method requires the obstacle boundaries to be parallel to the coordinate axes. Obstacles which are not aligned with the coordinate frame can be treated in the same way by rotating the coordinate system prior to the collision detection.

Line-Box Collision

Since the RRT algorithm expands its graph by adding nodes and connecting them to the tree, those connections have to be checked for collisions with obstacles. In the following, this collision test is done using straight lines between any two points which are connected in the RRT algorithm. Because an interpolation method is used on the points of the solution

path this is not sufficient in general. While the node points are fixed, the interpolation yields a differentiable connection which is a priori not known. Because the interpolation can only be done after the RRT solution is complete, it is assumed that the interpolated path is close to the straight line connection of edges in the graph.

Assumption 1 — *The interpolated path is sufficiently approximated by the line connecting two points of the solution path*

$$\boldsymbol{\sigma}(\lambda) \approx \mathbf{y}_k + c\,(\mathbf{y}_{k+1} - \mathbf{y}_k), \quad c \in [0;1],\ \lambda \in [\lambda_k; \lambda_{k+1}]\ . \tag{2.10}$$

Here $\mathbf{y}_k = \boldsymbol{\sigma}(\lambda_k)$ is the k-th node of the RRT. While this assumption may seem restrictive recall that there are two ways of influencing the relation (2.10): firstly, the distance between edges in the final path is statistically related to the number of samples generated for the RRT. A higher number of samples yields more points along which the path is interpolated. Thus, it can be used to control the deviation between the RRT planning and the interpolation. This was discussed in Section 2.4. Secondly, the maximum distance between two samples can be explicitly set in the RRT algorithm. Thus, the number of samples and the maximum distance determine the average distance between samples and have a significant influence on the interpolation. However, note that samples lying too close together can lead to oscillatory behaviour for polynomial interpolation. It is therefore important to tune the mentioned parameters to a characteristic dimension of the free space. Additionally, the minimum distance between the path and the obstacles in the environment can be increased by increasing the size of the obstacles. Note that in an Euclidean space the shortest connection of two points is a straight line. Thus, the use of an optimal RRT with respect to the path length converges to straight connections, see Section 2.4.1.

The collision detection of lines and approximated obstacles relies on a decomposition of the problem into one dimensional problems for each coordinate axis. To this end, the obstacle and the line are projected onto the coordinate axes. Similar to the point-box collision, obstacles not aligned with the coordinate frame are treated in a rotated coordinate frame. In general there are three cases: a disjoint projection, overlapping projections without a collision and, with a collision. For the first case, a collision is not possible, since a necessary condition is violated. Thus, there is no collision if along any axis the obstacle and the line segment are disjoint, i. e. the projections do not overlap. Figure 2.9 illustrates this situation, where the projection onto the coordinate axis y^1 is disjoint for the line and the box. Since this criterion is a necessary but not a sufficient condition it is used to dismiss a collision if the result is negative. The second and thereby the third case are discussed in the following.

For every collision it is trivially true that a point can be found such that the line and the box projections overlap in a projection onto any axis. This necessary condition is the key criterion of the following method. To specify this condition the parametric form of the line segment

$$\mathbf{l}(c) = \mathbf{y}_k + c(\mathbf{y}_{k+1} - \mathbf{y}_k), \quad c \in [0;1] \tag{2.11}$$

is used. With this definition the necessary condition can be reformulated for the given problem as follows. There is a collision if and only if there is a parameter value c^* such

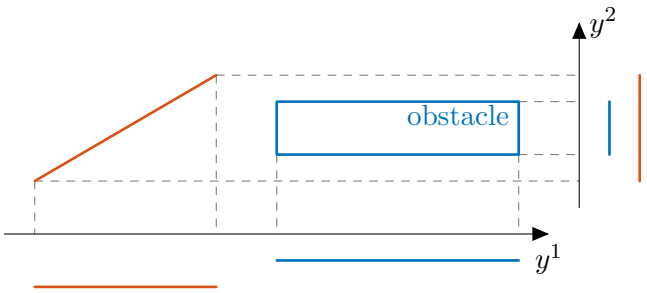

Figure 2.9: The line cannot intersect the box since its projection onto the y^1-axis does not overlap with the projection of the obstacle.

that the obstacle overlaps the line segment at least at one point $\mathbf{l}(c^*)$ in every projection. This concept is illustrated in Figure 2.10. Here, the projection of the line onto axis y^j is denoted by $l_{y^j}(c)$. The following method is used to check whether this condition is met

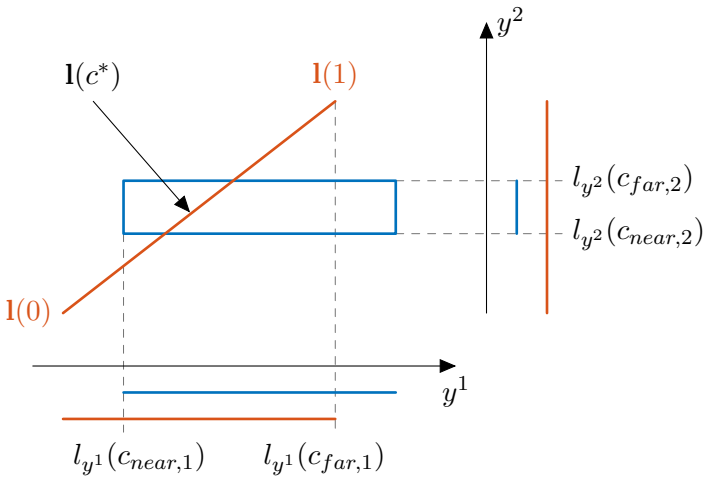

Figure 2.10: The line collides with the box, at each point $\mathbf{l}(c^*)$ with $c^* \in [c_{near,2}; c_{far,2}]$.

and thus a parameter value c^* exists. For clearness of exposition, the indices for each axis are omitted and vectors are treated component-wise.

To determine the existence of c^*, the two values c_{near} and c_{far} are calculated for each axis. These are the minimal and maximal parameter values, at which the line segment overlaps the box in a projection, respectively. Therefore, c_{near} can be interpreted as the parameter value where the line segment overlaps the edge of the box closer to the point $l(0)$. Likewise, $l(c_{far})$ corresponds to the projected edge closer to the point $l(1)$. If $l(0)$ or $l(1)$ lies between the minimal and maximal box value for the current axis, c_{near} or c_{far} are 0 or 1, respectively. These values are illustrated in Figure 2.10. Note that $\mathbf{l}(\max(c_{near}))$ is the point where the line intersects the box closer to its start and $\mathbf{l}(\min(c_{far}))$ is the point where the line intersects the box closer to its end. Furthermore, to exclude trivial

infinite cases, the parameter values 0 and 1 are used to identify no collision.

There cannot be a collision if the biggest value for c_{near} is bigger than the smallest value for c_{far} over all axes. That is because the relation $\max(c_{near}) > \min(c_{far})$ contradicts the necessary and sufficient condition that there exists a point such that there is an overlap for each axis. This fact can be intuitively explained by realising that c_{near} corresponds to the point at which the line intersects the near edge of the box projected onto some axis. The smaller value for c_{far}, however, states that at this parameter value the line segment has already passed the far edge of the obstacle in the projection onto another axis. Consequently, there cannot be a c^* for which the point along the line intersects the obstacle, and thus there cannot be a collision. The discussed relations are depicted in Figure 2.11. Whenever the conditions stated above are not met, there exists at least one

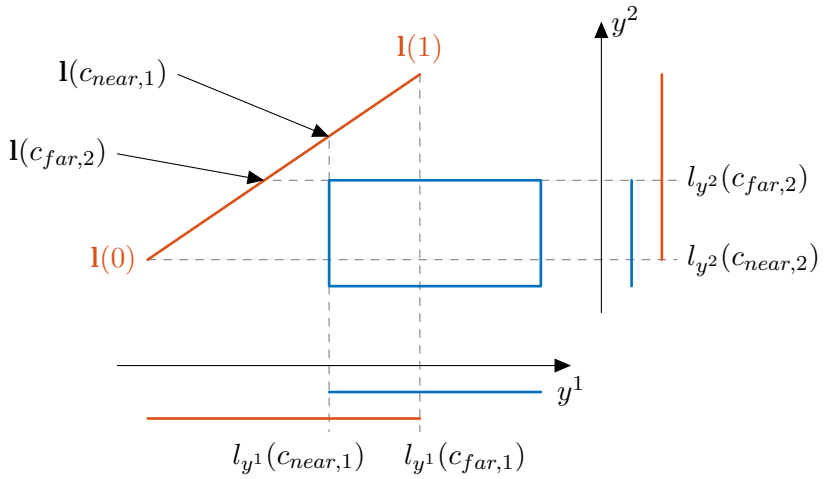

Figure 2.11: The line does not collide with the box, although their projections onto each axis overlap. From the distance of the respective points to $\mathbf{l}(0)$ it is clear that $\max(c_{near}) > \min(c_{far})$ holds.

c^* such that the projection of the corresponding point on the line intersects the obstacle. Considering that $\max(c_{near})$ determines the point where the line intersects the near edge of the box, the parameter values for the intersection have to be bigger or equal to this value. Similarly, they are smaller than or equal to $\min(c_{far})$ and thus every c^* comes to lie in the interval $[\max(c_{near}); \min(c_{far})]$. These observations are illustrated in Figure 2.12.

The parameter values c can be calculated after the projection onto an axis, since a projection is a linear map. From linearity it can be shown that

$$\mathbf{y} = c\,\mathbf{y}_{k+1} + (1-c)\,\mathbf{y}_k$$

implies that

$$\mathbf{P}\mathbf{y} = c\,\mathbf{P}\mathbf{y}_{k+1} + (1-c)\,\mathbf{P}\mathbf{y}_k$$

holds true, with the projection onto any axis $\mathbf{P}$. This proves that the projection preserves barycentres, and thus the parameter values c can indeed be calculated by using the

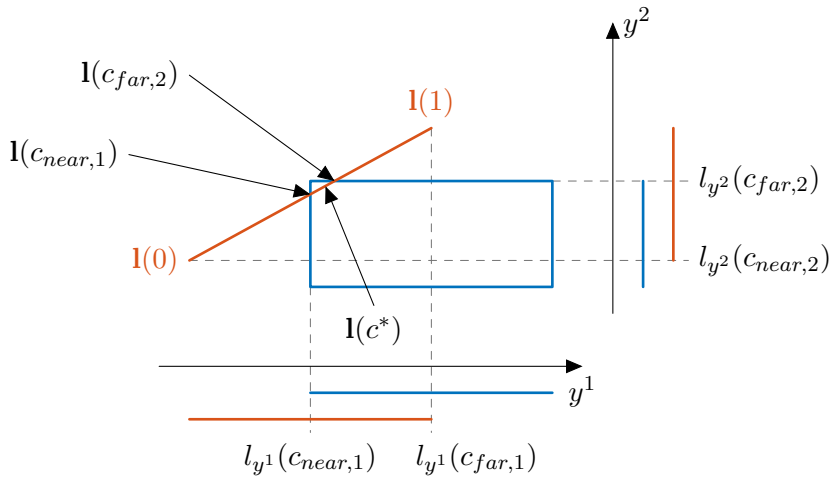

Figure 2.12: The line and box intersect, since the near edge is intersected at a smaller parameter value than the far edge, i. e. $c_{near,1} < c_{far,2}$. Every point $\mathbf{l}(c^*)$ with $c^* \in [c_{near,1}; c_{far,2}]$ is in the intersecting region.

projected points. Specifically, using a projection onto the y^1-coordinate this can be rewritten as

$$c_{near,1} = \frac{l_{y^1}(c_{near,1}) - l_{y^1}(0)}{l_{y^1}(1) - l_{y^1}(0)}$$

where the value $l_{y^1}(c_{near,1})$ is given as the y^1-coordinate of the box closer to $\mathbf{l}(0)$. Therefore, it has been shown that the calculation of the parameter value at the intersection can be decomposed into calculations on the real line.

2.5 Path Planning Results

In the following, after a brief discussion of the underlying assumptions, a Monte Carlo study of the proposed algorithm is presented.

The proposed approach relies on the RRT algorithm for searching for a feasible path in the free output space of the system. The system dynamics are neglected, which greatly simplifies collision detection. Consequently, it is not possible to calculate the inputs required to follow the solution path.

Because the system dynamics were neglected it is important to make sure that the system can follow the path by adjusting the parameters of the algorithm to avoid high curvatures. This is favourable to limit the actuation effort in sharp turns. Furthermore, the discrepancies of the reference path and the actual path followed by the system output can be relevant. This may lead to collisions, in particular for problems where the turning radius (at some nominal speed) and the characteristic lengths of the free space are of comparable size. A dynamically constrained turning radius is a problem mostly for systems with nonholonomic constrained dynamics. Since the focus of this work are mechanical systems, like robotic arms, it is assumed that the system can be forced to follow a path calculated

in its output space. Although relying on this approach can degrade the performance of the final trajectory, it is shown in simulations that the effect is negligible for the investigated systems. For nonholonomic systems, the system dynamics have to be included into the RRT algorithm to guarantee a feasible path.

To compare the results of the standard RRT to the optimal RRT a Monte Carlo simulation was performed. The search for the shortest path in 3-dimensional space between two fixed points was used as benchmark problem. The problem space is the unit cuboid obstructed by 10 cubic obstacles with an edge length of $a_c = 0.15$, uniformly distributed within the problem space. Obstacles directly obstructing the start or goal positions were resampled. The benchmark problem as solved with the optimal RRT is illustrated in Figure 2.13. Here, the initial position $\mathbf{y}_0 = (0.9, 0.9, 0.9)$ is marked by a

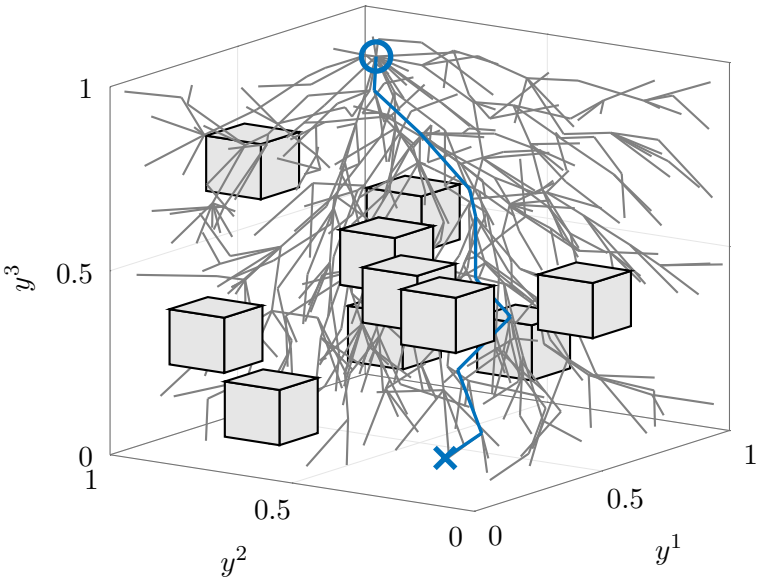

Figure 2.13: Example of a solution to the benchmark problem obtained from the optimal RRT with 500 nodes.

circle and the goal position $\mathbf{y}_{goal} = (0.1, 0.15, 0.1)$ by a cross.

For this problem, RRTs and optimal RRTs were generated with the same random samples, respectively. To this end the pseudo random number generator was initialised with the same seed for both methods. To judge the rate of convergence of the optimal RRT the algorithm was applied with the number of nodes varying from 100 to 2000 with an increment of 50 nodes. The simulation was repeated with 5000 optimal and regular RRTs. The statistical results of this Monte Carlo simulation can be found in Figure 2.14. The mean length of the path for the optimal RRT converges to the straight line connection denoted by J^*. However, since the obstacles randomly obscure this shortest path, it cannot be reached in general. The mean length of the path for the regular RRT does not significantly change with the number of samples. This is to be expected, since the first valid solution is not affected by further sampling. A noticeable outlier occurs with 100

nodes, since with this low number of samples the goal is not always reached. In this case, the path ending closest to the goal may not actually reach it and yields a short distance measure. The standard deviation of the cost function for the optimal case decreases similarly to the mean value. An optimal tree converges closer to the global optimum with an increasing number of samples, which reduces the statistical spread. The improved length of the path comes at the cost of computation time, as illustrated using the average computation times. Note that the regular RRT does not improve with the number of samples, as discussed before. Therefore, in a practical application it is terminated as soon as a feasible path is found. The difference in computation time comes from the two additional steps discussed in Section 2.4.1.

The simulation was performed on a 2.8GHz Intel Core i7 processor. The algorithms were implemented in ISO C99 and run as MATLAB Mex functions.

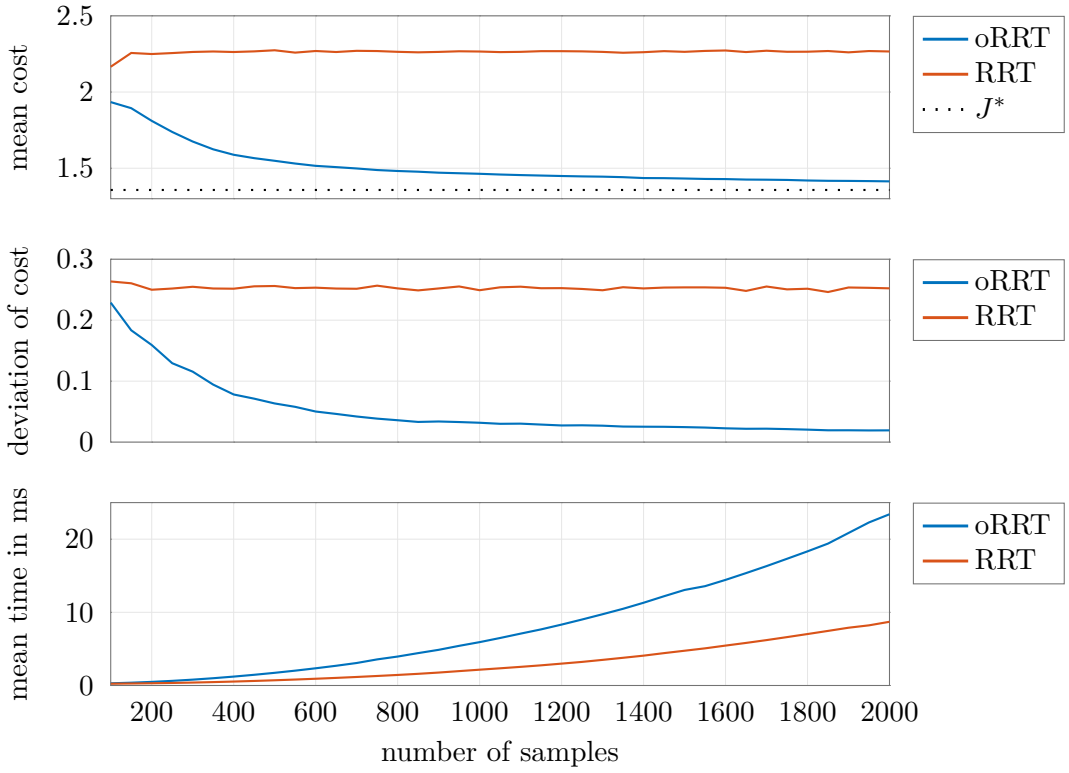

Figure 2.14: Statistical measures of 5000 optimal (oRRT) and regular (RRT) rapidly exploring random trees with a varying number of nodes.

3 Path Following Control

Driving the output of a system along a given path can can either be tackled as trajectory tracking or path following task. The former resembles an extension of set-point stabilisation methods by moving the set-point with a given time parametrisation. Path following can be regarded as generalisation to trajectory tracking without a priori time parametrisation. In other words, path following is concerned with driving the system output to a geometrical reference curve rather than to a fixed position at a fixed time.

Consider an autonomously driving car on a circular race track with a priori fixed time and position. If the car cannot follow the reference because of disturbances, like, for example heavy wind, it would move off the course on a short cut. Obviously, this behaviour is not desirable. A path following approach, however, would make the car stay on the track regardless of the feasibility of the desired speed. This behaviour is known as output invariance. Since the progression is accounted for along the path, the path following controlled car does not deviate from the track, though the desired velocity may not be achievable. This example illustrates the difference between the concept of trajectory tracking and path following.

A trajectory tracking controller drives the output of a system asymptotically to a desired reference trajectory, i. e. $\mathbf{y}(t) \to \mathbf{y}_{ref}(t)$ for $t \to \infty$. It is important to note that unlike for a path following controller, the reference trajectory $\mathbf{y}_{ref}(t)$ depends on time and a trajectory controller therefore depends on its a priori time parametrisation.

3.1 Literature Review

Despite not having a fixed time evolution, a goal of path following control is to attain a desired tangential movement along the path. A common method to achieve both output invariance and a tangential movement is to decouple tangential and transversal (normal) dynamics and control them individually. The independent control of the transversal subsystem is used to render the path invariant, thus the output of the system stays on the path in the nominal undisturbed case.

Banaszuk and Hauser [19] applied this feedback linearisation to the dynamics transversal to a periodic orbit. The transversal system, orthogonal to the reference orbit, is thereby rendered linear and can be easily controlled. Transversal feedback linearisation was generalised to embedded manifolds, see, e. g., Nielsen and Maggiore [20, 21]. Applications of this approach can be found, e. g., in [22, 23]. Therein, the path manifold is specified implicitly and the orthogonal dynamics are feedback linearised. A variety of control schemes were proposed for the resulting linear transversal system. Due to the need for an implicit curve representation self-intersecting curves are not admissible for this approach. Furthermore, the feasibility of feedback linearisation depends on the combination of path

and system. Finally, a second problem-specific control law has to be applied to move along the path.

Another path following strategy is to control the path parameter evolution of a parametrised curve with a trajectory controller. This allows for an evolution of a reference point on the path dependent on the state of the system, i. e. the time parameter of the trajectory controller depends on the system output. However, this approach does not guarantee output invariance. Combining the ideas of a time parametrisation and transversal feedback linearisation, Faulwasser [24] proposed a model predictive control scheme which does guarantee output invariance. Using this optimisation based feedback it is possible to systematically deal with constraints on the state or input of the system.

To systematically achieve a desired tangential movement along with a transversal feedback linearisation Consolini, Maggiore, *et al.* [25] introduced the notion of a closest point on the path. Using this point along the path, a moving reference frame corresponding to the output of the system can be introduced. This reference frame can be used to find a suitable tangential direction, which can be used to actuate the tangential dynamics. Using this approach, the requirements for a feedback linearisation are independent of the path. However, the main problem is to find orthogonal vectors spanning the transversal subsystem in dimensions higher than two. Gill, Kulić, *et al.* [26] rely on Frenet-Serret coordinates which are defined by the generalised curvature of the path. Even though this method uses parametrised curves, which allows for intersecting and piecewise defined paths, the curvature based frame limits its feasibility. Paths with vanishing or sign-switching curvature lead to ill-defined transversal coordinates. Bischof, Glück, *et al.* [27] introduced a novel coordinate frame based on parallel transport which overcomes this limitation.

A novel path following controller is presented in this thesis which combines the benefits of a parametrised curve representation and transversal feedback linearisation. Based on the approach Faulwasser used for model predictive path following a transversal formulation is defined in an orthonormal basis spanning the whole output space [24]. Similar to the ideas used in [28], the tangential problem is treated by an augmentation of the system. Combining these ideas and using suitable projections, a simple decomposition can be achieved in a static coordinate frame. Since this orthonormal basis is independent of the reference curve, the resulting approach has few requirements on the path and its continuity. Despite the good performance in a variety of simulations there is no systematic stability proof for this method. Furthermore, a parametrised path representation allows the use of a wide variety of curves with few assumptions, see Section 3.3.2. The resulting control law couples transversal and tangential subsystem to minimise the Euclidean norm of the error. Additionally, a modification of this control scheme is proposed which aims to decouple the virtual control inputs for the tangential and transversal subsystems. Extending the ideas of Flixeder, Glück, *et al.* [29], the resulting fixed frame of reference can be used to design a controller for each individual axis of the output space. For example, when machining anisotropic composite structures it is possible to define independent control laws parallel and orthogonal to the materials fibre-orientation while maintaining a different control strategy along the path. Similarly, if the path following concept is extended to include position and rotational degrees of freedom, these can be controlled separately.

3.2 Problem Statement

Subsequently, fully actuated rigid body mechanical systems which can be represented in the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \tag{3.1}$$

with the output function

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{3.2}$$

are considered. Therein, $\mathbf{x} \in \mathbb{R}^n$ is the system state, $\mathbf{y} \in \mathbb{R}^p$ the output and $\mathbf{u} \in \mathbb{R}^m$ the input. The vector fields $\mathbf{f}(\mathbf{x})$, $\mathbf{g}(\mathbf{x})$ and the p functions $\mathbf{h}(\mathbf{x})$ are assumed to be smooth, having continuous derivatives of all orders in their domain.

The main goal considered in this chapter is for the output of (3.2) to follow a parametrised curve

$$\boldsymbol{\sigma} : \Lambda \to \mathbb{R}^p \tag{3.3}$$

with the path parameter $\lambda \in \Lambda \subseteq \mathbb{R}$.

Assuming (3.2) yields a well defined relative degree at some point $\mathbf{x}_0$ for the system (3.1) static state feedback linearisation can be performed at this point. Therefore, a feedback transformation $\mathbf{u} = \mathbf{E}(\mathbf{x}) + \mathbf{D}(\mathbf{x})\mathbf{v}$ can be found in a neighbourhood U of $\mathbf{x}_0$ such that the system equations in the new $\boldsymbol{\xi}$-coordinates read as

$$\dot{\boldsymbol{\xi}} = \boldsymbol{\xi} + \mathbf{B}\,\mathbf{v}\ , \tag{3.4}$$

with the virtual control input $\mathbf{v}$ and the n by m input matrix $\mathbf{B}$. Additionally, there exists a diffeomorphism $T : \mathbf{x} \mapsto \boldsymbol{\xi}$, mapping the system state to the new coordinates. Further details are given in Section 3.3.1, for a rigorous treatment of feedback linearisation, see, e. g., [30].

The primary objectives for a path following controller can be summarised as follows, [31].

O1 Asymptotic convergence: For all initial conditions in a neighbourhood of the path, the output (3.2) asymptotically converges to the path, i. e. $\|\mathbf{y}(t) - \boldsymbol{\sigma}(\lambda(t))\| \to 0$ as $t \to \infty$ for some $\lambda \in \Lambda$.

O2 Output invariance: If the system state $\mathbf{x}(t_0)$ at some time t_0 is in an appropriate subset of $\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{h}(\mathbf{x}) = \boldsymbol{\sigma}\}$, this subset $\mathcal{P}^*$ is output invariant for the closed loop system, thus $\mathbf{x} \in \mathcal{P}^*,\ \forall t \geq t_0$.

O3 Tangential movement: An application specific movement along the path is achieved.

While there are other methods to have these or similar properties, the proposed approach tackles the problem in a full set of fixed coordinates. Furthermore, a wide range of reference curves can be used and the methodology can be easily applied to systems with arbitrary relative degree and output dimension.

3.3 Path Following using Feedback Linearisation

The path following control presented in this thesis satisfies the proposed objectives by separation of a tangential and transversal subsystem. An independent asymptotically stable transversal subsystem identically meets **O1**. The difference between trajectory control and path following is the explicit guarantee of an invariant set which is a subset of the path, i. e. **O2**. To meet **O3** it has to be possible to control the tangential movement along the path. Since the approach uses feedback linearisation, the basic notion of this nonlinear feedback approach is given in Section 3.3.1.

The main idea of the presented algorithm is based on the decomposition of the system into a tangential and transversal subsystem. The approach can be summarised in the following steps.

1. The tangential subsystem is stated by introducing the notion of a tangential state in Section 3.3.3. This state is the part of the output specifying the position along the path. The tangential subspace has the dimension of the path.

2. The transversal space is comprised of the error to the closest point on the path. It spans the complementary space to the tangential space. The transversal subspace is formulated in terms of a basis redundantly spanning the output space.

3. Feedback linearisation is applied to both subsystems in Section 3.3.4 yielding two redundant linear time invariant (LTI) systems. The redundancy is solved by using suitable projections.

4. The tangential and transversal subsystems are individually controlled to meet the specified goals in Section 3.3.5.

It is shown in Section 3.3.4 that with the approach presented in this thesis it is not possible to decouple the tangential and transversal subsystems perfectly. Yet the coupling is shown to be negligible. Two concepts for the path following controller are proposed, one of which couples the subsystems by minimizing the combined transversal and tangential error, the other based on a projection.

3.3.1 Feedback Linearisation

To introduce the notation and facilitate the following calculations, the methodology of static state feedback linearisation is discussed. After stating the equations describing a system in a general fashion, the time derivative of the output is expressed in terms of Lie derivatives. This formulation is then used to construct a feedback law which performs an exact linearisation of the system.

In the following, fully actuated rigid body mechanical systems of the form (3.1) with the output function (3.2) are considered. The time derivative of the output is given by

$$\dot{\mathbf{y}} = \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right) \dot{\mathbf{x}} = \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right) \mathbf{f}(\mathbf{x}) + \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right) \mathbf{g}(\mathbf{x})\ \mathbf{u} = \mathrm{L}_{\mathbf{f}}\mathbf{h} + \mathrm{L}_{\mathbf{g}}\mathbf{h}\ \mathbf{u}\ , \tag{3.5}$$

where the Lie derivative of the output along the vector field $\mathbf{f}(\mathbf{x})$ is defined as

$$\mathrm{L}_{\mathbf{f}}\mathbf{h} := \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right)\mathbf{f}(\mathbf{x}) \ .$$

Note that by a slight abuse of notation the Lie derivative is directly applied to the column vector of functions $\mathbf{h}$, which is defined as component-wise application of the Lie derivative. Furthermore, the Lie derivative along multiple vector fields $\mathrm{L}_{\mathbf{g}}\mathbf{h}$ is defined as the matrix comprised of column vectors of the Lie derivatives along each vector field. Thereby, the time derivative of the output of order r yields

$$\mathbf{y}^{(r)} = \mathrm{L}_{\mathbf{f}}^{r}\mathbf{h} + \mathrm{L}_{\mathbf{g}}\mathrm{L}_{\mathbf{f}}^{r-1}\mathbf{h}\,\mathbf{u} \ , \tag{3.6}$$

where higher order Lie derivatives are defined recursively by

$$\mathrm{L}_{\mathbf{f}}^{k}\mathbf{h} := \mathrm{L}_{\mathbf{f}}\left(\mathrm{L}_{\mathbf{f}}^{k-1}\mathbf{h}\right) \ , \quad \mathrm{L}_{\mathbf{f}}^{0}\mathbf{h} = \mathbf{h}(\mathbf{x}) \ .$$

The autonomous dynamics of order r of the output are subsequently denoted by

$$\mathbf{E} := \mathrm{L}_{\mathbf{f}}^{r}\mathbf{h} \ .$$

With the definition of the decoupling matrix

$$\mathbf{D} := \mathrm{L}_{\mathbf{g}}\mathrm{L}_{\mathbf{f}}^{r-1}\mathbf{h} \ , \tag{3.7}$$

the relative degree of the system can be defined as follows.

Definition.—A system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\,\mathbf{u}$ with $\mathbf{u} \in \mathbb{R}^m$, and output $\mathbf{y} = \mathbf{h}(\mathbf{x}) \in \mathbb{R}^p$, with $m \geq p$, has vector relative degree $\{r_1, \ldots, r_p\}$ at a point $\mathbf{x}_0$ if

1. $\mathrm{L}_{g_j}\mathrm{L}_{\mathbf{f}}^{k}h_i(\mathbf{x}) = 0$, for all $j = 1, \ldots, m$, $i = 1, \ldots, p$, $k < r_i - 1$ and for all $\mathbf{x}$ in a neighbourhood of $\mathbf{x}_0$, where $\mathbf{g}_j$ are input vector fields.

2. The $p \times m$ decoupling matrix

$$\mathbf{D} = \begin{bmatrix} \mathrm{L}_{\mathbf{g}_1}\mathrm{L}_{\mathbf{f}}^{r_1-1}h_1(\mathbf{x}) & \ldots & \mathrm{L}_{\mathbf{g}_m}\mathrm{L}_{\mathbf{f}}^{r_1-1}h_1(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \mathrm{L}_{\mathbf{g}_1}\mathrm{L}_{\mathbf{f}}^{r_p-1}h_p(\mathbf{x}) & \ldots & \mathrm{L}_{\mathbf{g}_m}\mathrm{L}_{\mathbf{f}}^{r_p-1}h_p(\mathbf{x}) \end{bmatrix}$$

   has full row rank p at $\mathbf{x} = \mathbf{x}_0$.

According to this definition, if (3.7) has full rank p at a point $\mathbf{x}_0$ and the lower order Lie derivatives vanish, the system (3.1) with output (3.2) has a well defined vector relative degree in the neighbourhood of this point. It is therefore locally feedback linearisable.

Assumption 2 — *The system has a locally well defined relative degree and is therefore feedback linearisable.*

The aim of feedback linearisation is to control the nonlinear system (3.1) in a way such that it behaves like a linear system. Because any state space representation of a linear system is equivalent under linear transformation, the resulting linear system is chosen to be a simple integrator chain. This desired system has the form

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{y}^{(1)} \\ &\vdots \\ \dot{\mathbf{y}}^{(r-1)} &= \mathbf{y}^{(r)} \\ \mathbf{y}^{(r)} &= \mathbf{v} \ , \end{aligned}$$

with the virtual control input $\mathbf{v}$. To achieve this linear system behaviour, the nonlinear terms have to be compensated by the physical system input. The control input satisfying this condition can be found by substituting Equation (3.6), which yields

$$\mathbf{y}^{(r)} = \mathbf{E} + \mathbf{D}\,\mathbf{u} = \mathbf{v}$$

and solving for the system input $\mathbf{u}$. Therefore, the control input to the system is determined by the nonlinear feedback

$$\mathbf{u} = \mathbf{D}^{-1}\left(\mathbf{v} - \mathbf{E}\right) \ . \tag{3.8}$$

Note that the inverse of the decoupling matrix exists since by Assumption 2 it has full rank at least locally. If the number of inputs is larger than the number of outputs $m > p$ the inverse has to be replaced by the Moore-Penrose pseudo-inverse.

For systems not satisfying Assumption 2, dynamic feedback linearisation can be applied. Using a dynamic extension of the controller, the system is altered to have a well defined relative degree. Furthermore, systems with a deficient relative degree, i. e. where $\sum_i r_i = n$ is not satisfied, can be treated by output feedback linearisation. Both of these generalisations can be used along with the methods proposed in this thesis. For further details, see [30].

3.3.2 Admissible Paths

In the following, assumptions on the path and its required formulation are discussed. A general description of a curve can be given by piecewise functions

$$\boldsymbol{\sigma}_k : \mathbb{R} \to \mathbb{R}^p \,, \quad k \in \{1, 2, \ldots, n_{seg}\}$$

together with n_{seg} nonempty closed intervals on the real line $\Lambda_k := [\lambda_{k,start}, \lambda_{k,end}] \subset \mathbb{R}$. The path is expressed as the union of the sets defined by each curve segment

$$\mathcal{P} := \bigcup_{k=1}^{n_{seg}} \boldsymbol{\sigma}_k(\Lambda_k) \ .$$

To simplify the notation, the subscript k is omitted even though all derivations and results hold for piecewise defined curves. The curve can be open or closed by having an open or closed first and or last set Λ_k. Both of these cases are admissible sets of path parameter values.

In order for a tangent direction to be defined it is assumed that:

Assumption 3 — *The path is regular i. e.* $\boldsymbol{\sigma}' \neq \mathbf{0} \;\; \forall\, \lambda \in \Lambda$.

Furthermore, to guarantee a continuous feedback law the target path has to be sufficiently smooth.

Assumption 4 — *The path is continuously differentiable up to the order of one plus the maximum relative degree of the system, i. e.* $\boldsymbol{\sigma} \in \mathcal{C}^{\hat{r}+1}$ *with* $\hat{r} = \max r_i$, $i = 1, \ldots, p$ *the maximum relative degree of* (3.1).

Assumption 4 is used to guarantee a continuous feedback law for a given system.

3.3.3 Tangential and Transversal Dynamics

In this section, the notion of tangential and transversal coordinates is developed. First, the path parameter is specified and the corresponding arc length of the path is introduced. The time derivative of the path parameter is consequently used as tangential system along with the arc length as tangential output. The distance between the system output and the path is defined as transversal coordinate. This tangential and transversal output will be used in Section 3.3.4 to obtain a system in the form (3.4) by state feedback linearisation.

With the parametric path representation of Section 3.3.2 it is straightforward to identify the unit tangent vector

$$\mathbf{e}^{\parallel} := \frac{\boldsymbol{\sigma}'(\lambda)}{\|\boldsymbol{\sigma}'(\lambda)\|}$$

by the partial derivative $(\cdot)'$ of the curve with respect to the path parameter λ. Since the tangent vector depends on the path parameter, a value λ^* corresponding to the current output of the system has to be defined. A straightforward choice for the path parameter is the projection of the system output onto the path. Therefore, similar to Consolini, Maggiore, *et al.* [25] the projection operator ϖ is introduced. This function determines the path parameter corresponding to the closest point on the path according to the Euclidean norm

$$\varpi(\mathbf{y}) := \arg\min_{\lambda \in \Lambda} \|\mathbf{y} - \boldsymbol{\sigma}(\lambda)\|^2 \; .$$

Since in the present context only the Euclidean norm is used, it is not explicitly specified. In the following, it is assumed that λ fulfils this equation and thus

$$\lambda^* = \varpi \circ \mathbf{h}(\mathbf{x}) \tag{3.9}$$

holds true, where $\circ$ is the composition of functions. Following the ideas of Flixeder, Glück, *et al.* [29], the first order optimality condition of (3.9) is given by

$$-\left(\mathbf{y} - \boldsymbol{\sigma}\right)^{\mathrm{T}} \boldsymbol{\sigma}' = 0 \; . \tag{3.10}$$

Note that this holds true only if λ^* is on the inside of Λ or the latter is an open set. For continuity reasons this has to be the case between each curve segment. At the starting- and endpoint of a closed set one has to make sure that violating this condition does not lead to unexpected behaviour. The second order optimality condition

$$(\boldsymbol{\sigma}')^{\mathrm{T}}\boldsymbol{\sigma}' - (\mathbf{y}-\boldsymbol{\sigma})^{\mathrm{T}}\ \boldsymbol{\sigma}'' > 0$$

is necessary and sufficient for a minimum. This equation can be used to find the tubular neighbourhood of the path for which a unique closest path parameter can be found using (3.9), see, e. g., [32]. By taking the time derivative of (3.10)

$$-(\boldsymbol{\sigma}')^{\mathrm{T}}\left(\dot{\mathbf{y}} - \boldsymbol{\sigma}'\dot{\lambda}^*\right) - (\mathbf{y}-\boldsymbol{\sigma})^{\mathrm{T}}\ \boldsymbol{\sigma}''\,\dot{\lambda} = 0$$

and reordering terms, an expression for $\dot{\lambda}^*$ can be found, namely

$$f_\lambda(\lambda^*, \mathbf{x}) := \dot{\lambda}^* = \frac{(\boldsymbol{\sigma}')^{\mathrm{T}}}{\|\boldsymbol{\sigma}'\|^2 - (\mathbf{y}-\boldsymbol{\sigma})^{\mathrm{T}}\ \boldsymbol{\sigma}''}\ \dot{\mathbf{y}}\ . \tag{3.11}$$

Introducing the scalar factor

$$\kappa := \frac{1}{\|\boldsymbol{\sigma}'\|^2 - (\mathbf{y}-\boldsymbol{\sigma})^{\mathrm{T}}\ \boldsymbol{\sigma}''}\quad ,$$

Equation (3.11) can be rewritten as

$$\dot{\lambda}^* = \kappa\ (\boldsymbol{\sigma}')^{\mathrm{T}}\ \dot{\mathbf{y}}\ .$$

The scaling factor κ depends on the distance of the output to the closest point on the path and the curvature of the path. Comparing the definition of κ to the second order optimality condition, it is easy to see that the projection (3.9) has a unique solution whenever κ is nonsingular. Such a singularity can only occur if the output is on the concave side of the path, i. e. $\boldsymbol{\sigma}'' > 0$, which can be inferred from the definition of κ. These singular points form the closure of a tubular neighbourhood within which a unique path parameter can be found. Outside of this region, another point along the path is closer to the current output and the value of λ^* is discontinuous. Note, however, that this is a local observation and in general not valid for the entire path.

These notions are illustrated for a circular path in Figure 3.1. Here, the factor κ is drawn as the third axis to a two-dimensional path. Its value diverges in the proximity to the centre of the circle. On the path, κ equals the inverted square norm of the tangent vector. While on the convex outside κ approaches zero. If the output of the system moves beyond the centrepoint, κ is discontinuous. That is, because a point on the other half circle is the new closest point. Intuitively, a high value of κ is an indicator for an ambiguity of the closest point on the path.

Note that a solution curve of Equations (3.1) and (3.2) is at least $\hat{r}$-times continuously differentiable. Therefore, $\boldsymbol{\sigma} \in \mathcal{C}^{\hat{r}}$ is necessary for the system to follow the path. The time derivative of the path parameter was derived by using the optimality condition (3.10). Thus, this method requires one path derivative more than $\hat{r}$, which is presumed in Assumption 4.

By the definition of the path parameter given in Equation (3.9), a dynamical system can be defined corresponding to the tangential movement. For curves parametrised by their arc length, i. e. $\|\boldsymbol{\sigma}'\| = 1\,,\ \forall\lambda$, the output of this tangential system is chosen to be

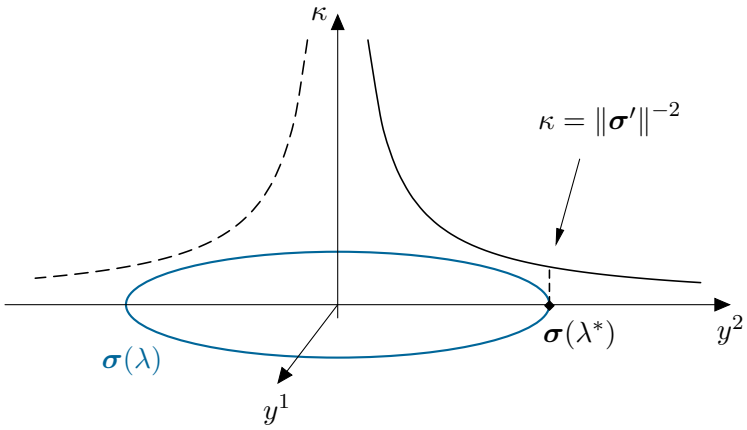

Figure 3.1: Orthogonal to a circular path, κ increases closer to the centrepoint, where it is singular. This singular point yields an ambiguous closest point. On the outer convex side, it decreases asymptotically to zero.

the path parameter $\xi^{\parallel} = \lambda^*$. To generalise the treatment to curves that are not unit-speed parametrised, the arc length

$$l(\mathbf{y}) := \int_{\lambda_0}^{\lambda^*} \|\boldsymbol{\sigma}'(\zeta)\| \, \mathrm{d}\zeta$$

between the fixed initial parameter value λ_0 and the projected path parameter λ^* is introduced. With $l(\cdot)$, the position along the path can be expressed as arc length between a reference point λ_0 and the current path parameter value. The stated formula implicitly requires continuity between the sets Λ_k. This assumption is made to further simplify the notation. For general sets Λ_k, the summation over all contributions of l along each segment k has to be used instead. The arc length of the traversed path defines the tangential progression

$$\xi^{\parallel} := l \circ \mathbf{h}(\mathbf{x}) \ .$$

Next, a suitable transversal coordinate, which is orthogonal to the tangential one, is defined. To this end, recall that the output space $\mathbb{R}^p$ is spanned by the tangential vector and its orthogonal hyperplane. Hence, the transversal vectors lie within this hyperplane. Using Equation (3.10) and Assumption 3 it is straightforward to see that

$$\boldsymbol{\xi}^{\perp} := \mathbf{y} - \boldsymbol{\sigma}(\lambda^*)$$

is orthogonal to the tangent vector. Hence each vector in the output space can be identified by a position along the path and the orthogonal distance to this point, i. e. the two components $\boldsymbol{\xi} = [\xi^{\parallel}, \boldsymbol{\xi}^{\perp}]^{\mathrm{T}}$. Note that $\boldsymbol{\xi} \in \mathbb{R}^{p+1}$ is a redundant parametrisation of the p-dimensional output space, since $\xi^{\parallel} \in \mathbb{R}$ and $\boldsymbol{\xi}^{\perp} \in \mathbb{R}^p$.

3.3.4 Path Feedback Linearisation

In the following section, feedback linearisation is applied to the dynamics of the tangential and transversal coordinates introduced in Section 3.3.3. First, the dynamical system

(3.1) is augmented by the dynamics of the path parameter (3.11). The output of this constructed system is chosen to be the tangential and transversal coordinates $(\xi^{\parallel}, \boldsymbol{\xi}^{\perp})$. Using this augmented system, feedback linearisation can be applied to the tangential and transversal coordinates.

The original system (3.1) is augmented by (3.11) yielding

$$\begin{aligned} \dot{\mathbf{x}}_A &= \mathbf{f}_A(\mathbf{x}_A) + \mathbf{g}_A(\mathbf{x})\ \mathbf{u} = \\ \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\lambda^* \\ \mathbf{x}\end{bmatrix} &= \begin{bmatrix} f_\lambda(\lambda^*, \mathbf{x}) \\ \mathbf{f}(\mathbf{x})\end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{g}(\mathbf{x})\end{bmatrix}\ \mathbf{u}\ , \end{aligned} \tag{3.12}$$

with the augmented state $\mathbf{x}_A = [\lambda, \mathbf{x}]^{\mathrm{T}}$, the augmented system dynamics $\mathbf{f}_A(\mathbf{x}_A) = [f_\lambda(\mathbf{h}(\mathbf{x})), \mathbf{f}(\mathbf{x})]^{\mathrm{T}}$, and $\mathbf{g}_A(\mathbf{x}_A) = [0, \mathbf{g}(\mathbf{x})]^{\mathrm{T}}$. Note that $f_\lambda(\lambda^*, \mathbf{x})$ depends on the projected path parameter as defined in Section 3.3.3. The augmented output function $\mathbf{h}_A(\mathbf{x})$ is composed of the tangential and transversal output

$$\mathbf{y}_A = \mathbf{h}_A(\mathbf{x}_A) = \begin{bmatrix}\xi^{\parallel} \\ \boldsymbol{\xi}^{\perp}\end{bmatrix}\ . \tag{3.13}$$

To simplify the treatment of the nonlinear system (3.12) feedback linearisation similar to Section 3.3.1 is used. With the virtual output (3.13) this can be understood as the exact linearisation of the tangential movement and the transversal distance. The time derivatives of the output are

$$\dot{\mathbf{y}}_A = \mathrm{L}_{\mathbf{f}_A}\mathbf{h}_A + \underbrace{\mathrm{L}_{\mathbf{g}_A}\mathbf{h}_A}_{=\mathbf{0}}\mathbf{u}$$

and

$$\ddot{\mathbf{y}}_A = \mathrm{L}^2_{\mathbf{f}_A}\mathbf{h}_A + \mathrm{L}_{\mathbf{g}_A}\mathrm{L}_{\mathbf{f}_A}\mathbf{h}_A\ \mathbf{u} = \mathbf{E}_A(\mathbf{x}_A) + \mathbf{D}_A(\mathbf{x}_A)\ \mathbf{u}\ , \tag{3.14}$$

with the decoupling matrix $\mathbf{D}_A(\mathbf{x}_A)$ and the autonomous dynamics $\mathbf{E}_A(\mathbf{x}_A)$ of the augmented system. Details on the derivation of these terms can be found in Appendix A. Note that the projected path parameter λ^* is actuated by the system input and thus has the same relative degree as the physical system (3.1). Therefore, the augmented system has full relative degree and state feedback linearisation can be applied to this system. As a consequence of the augmentation by the redundant $\boldsymbol{\xi}$ coordinates, a diffeomorphism cannot be found between the augmented states and the original system states. Informally, the input

$$\mathbf{u} = \left(\mathbf{D}_A(\mathbf{x}_A)\right)^{-1}\ \left(\mathbf{v} - \mathbf{E}_A(\mathbf{x}_A)\right) \tag{3.15}$$

using the new virtual input $\mathbf{v} = [v^{\parallel}, \mathbf{v}^{\perp}]^{\mathrm{T}}$ transforms the system into the form (3.4). Thereby $v^{\parallel}$ is designed to control the tangential and $\mathbf{v}^{\perp}$ the transversal dynamics. Since the inverse of the augmented decoupling matrix is not defined appropriate substitutions are presented in the following.

Further investigating the decoupling matrix $\mathbf{D}_A(\mathbf{h}_A(\mathbf{x}_A))$ shows that it can be factorised in the form

$$\mathbf{D}_A(\mathbf{h}_A(\mathbf{x}_A)) = \mathbf{K}(\mathbf{h}_A(\mathbf{x}_A))\ \mathbf{D}(\mathbf{h}(\mathbf{x}))$$

with the decoupling matrix of the original system (3.1)

$$\mathbf{D}(\mathbf{h}(\mathbf{x})) = \mathrm{L}_{\mathbf{g}}\mathrm{L}_{\mathbf{f}}\mathbf{h}(\mathbf{x})$$

and the matrix

$$\mathbf{K}(\mathbf{h}_A(\mathbf{x}_A)) = \begin{bmatrix} \|\boldsymbol{\sigma}'\|\ \kappa\ (\boldsymbol{\sigma}')^{\mathrm{T}} \\ \mathbf{I}_{p\times p} - \kappa\ \boldsymbol{\sigma}'\ (\boldsymbol{\sigma}')^{\mathrm{T}} \end{bmatrix} \ . \tag{3.16}$$

By Assumption 2 the decoupling matrix $\mathbf{D}(\mathbf{h}(\mathbf{x}))$ has full rank p. In the augmented system, however, the input is divided into two subspaces by the first row and the last p rows of $\mathbf{K}$, as desired. The augmented decoupling matrix $\mathbf{D}_A(\mathbf{h}_A(\mathbf{x}_A))$ thus maps the input to the tangential and transversal coordinates. The first row of the matrix (3.16) is a projection onto the tangential vector. The tangential subspace has dimension 1 for all $\lambda \in \Lambda$ as per Assumption 3, under which the path is regular. The transversal components of the output are mapped by the matrix

$$\mathbf{P}_\kappa := \mathbf{I}_{p\times p} - \kappa\ \boldsymbol{\sigma}'(\boldsymbol{\sigma}')^{\mathrm{T}} \ .$$

There are two cases to be distinguished:

- If $\kappa = \|\boldsymbol{\sigma}'\|^{-2}$ (i. e. the output is on the path or the path has locally vanishing curvature) the given matrix is

  $$\mathbf{P} := \mathbf{I}_{p\times p} - \frac{\boldsymbol{\sigma}'(\boldsymbol{\sigma}')^{\mathrm{T}}}{\|\boldsymbol{\sigma}'\|^2} \ .$$

  This matrix can be identified as the orthogonal complement of the projection onto the tangential space. Hence it forms the projection into the transversal subspace of $\mathbb{R}^p$. This projection decouples the transversal from the tangential dynamics.

- In any other case the transversal matrix has full rank p and is no projection matrix. Therefore, any change of the output changes the transversal error to the path. Figure 3.2 shows the general situation, where clearly the curvature of the path and the error to the path result in transversal changes for any change of the output.

For a more detailed analysis, see Appendix B. To invert the rectangular matrix $\mathbf{K}$ in (3.15) the Moore-Penrose pseudo-inverse, denoted by $(\cdot)^\dagger$, is used. The resulting feedback law

$$\mathbf{u} = \left(\mathbf{D}(\mathbf{x})\right)^{-1}\ \mathbf{K}^\dagger\ \left(\mathbf{v} - \mathbf{E}_A(\mathbf{x}_A)\right) \tag{3.17}$$

leads to a fast convergence to the path at the cost of a deviation from the tangential reference. This pseudo-inverse controller solves the feedback linearisation problem optimal with respect to the Euclidean norm. Practically, in significant distance to the path the transversal error is prioritised over the tangential tracking error. A detailed discussion of the performance of this method can be found in Chapter 4.

For some applications, this coupling may be an undesired behaviour. To achieve an independent tangential controller, the pseudo-inverse of the matrix $\mathbf{K}$ can be replaced by

$$\tilde{\mathbf{K}} := \begin{bmatrix} \dfrac{\boldsymbol{\sigma}'}{\kappa\|\boldsymbol{\sigma}'\|^3} & \mathbf{P} \end{bmatrix} \ ,$$

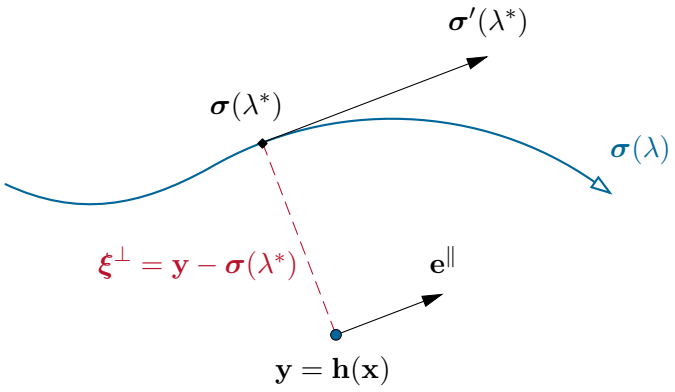

Figure 3.2: Since the output $\mathbf{y}$ is not on the path a tangential movement causes a change in the transversal distance $\boldsymbol{\xi}^\perp$ due to the curvature of the path.

which yields the decoupling control law

$$\mathbf{u} = \left(\mathbf{D}(\mathbf{x})\right)^{-1} \tilde{\mathbf{K}} \left(\mathbf{v} - \mathbf{E}_A(\mathbf{x}_A)\right) \ . \tag{3.18}$$

This method is based on the projection of the virtual input onto the tangent vector and into the normal plane, respectively. If $\kappa = \|\boldsymbol{\sigma}'\|^{-2}$, in particular along the path, the control laws (3.17) and (3.18) are equivalent. For a general point in the output space, substitution of the input (3.18) into the augmented system (3.14) yields

$$\begin{bmatrix} \ddot{\xi}^\parallel \\ \ddot{\boldsymbol{\xi}}^\perp \end{bmatrix} = \mathbf{E}_A + \begin{bmatrix} \|\boldsymbol{\sigma}'\| \ \kappa \ (\boldsymbol{\sigma}')^{\mathrm{T}} \\ \mathbf{P}_\kappa \end{bmatrix} \begin{bmatrix} \dfrac{\boldsymbol{\sigma}'}{\kappa \|\boldsymbol{\sigma}'\|^3} & \mathbf{P} \end{bmatrix} (\mathbf{v} - \mathbf{E}_A)$$

where $\mathbf{D}\mathbf{D}^{-1} = \mathbf{I}_{p\times p}$ was omitted for brevity. By using the relations of Appendix B on projections, the system can be described by the differential equations

$$\begin{bmatrix} \ddot{\xi}^\parallel \\ \ddot{\boldsymbol{\xi}}^\perp \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}_{1\times p} \\ \boldsymbol{\varepsilon} & \mathbf{P} \end{bmatrix} \begin{bmatrix} v^\parallel \\ \mathbf{v}^\perp \end{bmatrix} + \begin{bmatrix} 0 \\ \left(\mathbf{I}_{p\times p} - \mathbf{P}\right) \mathrm{L}^2_{\mathbf{f}_A} \boldsymbol{\xi}^\perp - \boldsymbol{\varepsilon} \ \mathrm{L}^2_{\mathbf{f}_A} \xi^\parallel \end{bmatrix} \ . \tag{3.19}$$

The coupling of the transversal dynamics is given by

$$\boldsymbol{\varepsilon} = \mathbf{P}_\kappa \frac{\boldsymbol{\sigma}'}{\kappa \|\boldsymbol{\sigma}'\|^3} \ .$$

Recall that $\mathbf{P}$ is the projection into the normal plane, which is orthogonal to the tangent vector. Its complementary matrix $\mathbf{I}_{p\times p} - \mathbf{P}$ projects into the tangential space. If $\kappa = \|\boldsymbol{\sigma}'\|^{-2}$, especially along the path, the coupling term $\boldsymbol{\varepsilon}$ is proportional to $\mathbf{P}\boldsymbol{\sigma}' = \mathbf{0}$, because $\mathbf{P}_\kappa = \mathbf{P}$. In this case, the coupling vanishes and the dynamics degenerate to

$$\begin{aligned} \ddot{\xi}^\parallel &= v^\parallel \\ \ddot{\boldsymbol{\xi}}^\perp &= \mathbf{v}^\perp + \left(\mathbf{I}_{p\times p} - \mathbf{P}\right) \mathrm{L}^2_{\mathbf{f}_A} \boldsymbol{\xi}^\perp \ . \end{aligned} \tag{3.20}$$

Therefore, the tangential and transversal subsystem can be controlled independently using $v^{\parallel}$ and $\mathbf{v}^{\perp}$ along the path. Note that it was assumed that the virtual input $\mathbf{v}^{\perp}$ is chosen to lie in the transversal subspace and therefore $\mathbf{P}\mathbf{v}^{\perp} = \mathbf{v}^{\perp}$. On a general point in the output space, the curvature couples each movement with the transversal error. However, extensive simulation studies have shown that these dynamics are negligible compared to the transversal control authority, see Chapter 4. Furthermore, it is worth noting that in other methods presented in literature this coupling of error and curvature is avoided by the choice of a moving coordinate frame attached to the output. By projection of the error onto a local coordinate frame, the coupling with the tangential acceleration term vanishes, see, e. g., [26, 27, 29].

The nonlinear feedback law (3.15) provides means to directly influence the tangential and transversal system. For the control inputs $v^{\parallel}$ and $\mathbf{v}^{\perp}$ independent controllers can be designed for both subsystems. Since they act on the simple system (3.20), for these control laws a wide variety of control strategies is available.

3.3.5 Virtual Control Strategy

The simplified systems (3.20) has to be controlled to render the path following controller stable. This is the main focus of the following section. Furthermore, it is checked by linear control theory, whether the control law (3.15) in combination with a suitable virtual control law satisfies the objectives stated in Section 3.2. The virtual control law used in this context consists of two parts, a feedback and a feedforward term. A PID controller is used as feedback control law. This feedback is combined with a feedforward term to achieve a fast tracking of the reference velocity along the path. This feedforward term is realised using a reference filter.

Reference Filter

To achieve a fast convergence to the reference position, while retaining a limited velocity, the given reference position is filtered. This filter yields a reference velocity and acceleration as well as a time parametrised position.

The tangential reference position, velocity and acceleration are obtained using a two stage reference filter. The desired position along the arc length $\xi_d^{\parallel}$ is limited by a rate limiter $\dot{\xi}_l^{\parallel} \leq \dot{\xi}_{max}^{\parallel}$ with the maximal tangential velocity $\dot{\xi}_{max}^{\parallel}$ and the rate limited reference arc length $\xi_l^{\parallel}$. Note that the limit on the tangential reference does not impose a hard limit on the velocity of the output.

The rate-limited reference is then filtered by a linear third order filter, which is used not only to smooth the signal, but also to obtain a reference velocity and acceleration. Let s be the Laplace variable, then the filter is given by

$$\begin{bmatrix} \xi_r^{\parallel} \\ \dot{\xi}_r^{\parallel} \\ \ddot{\xi}_r^{\parallel} \end{bmatrix} = \begin{bmatrix} 1 \\ s \\ s^2 \end{bmatrix} \frac{a_0}{s^3 + a_2\, s^2 + a_1\, s + a_0}\ \xi_l^{\parallel} \ . \tag{3.21}$$

One pole p with multiplicity three is chosen which yields $a_0 = -p^3$, $a_1 = 3p^2$ and $a_2 = -3p$.

The structure of the whole input filter is depicted in Figure 3.3.

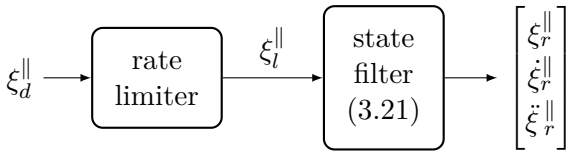

Figure 3.3: Block diagram of the filter used to achieve a smooth and velocity limited reference and the corresponding derivatives.

Virtual Control Law

Relying on the feedback linearisation control (3.15) the virtual control law can be chosen as

$$\begin{aligned} v^{\parallel} &= \ddot{\xi}_r^{\parallel} - k_d^{\parallel}\,\dot{e}^{\parallel} - k_p^{\parallel}\,e^{\parallel} - k_i^{\parallel}\int_0^t e^{\parallel}(\tau)\,\mathrm{d}\tau \\ \mathbf{v}^{\perp} &= -\mathbf{k}_d^{\perp}\,\dot{\boldsymbol{\xi}}^{\perp} - \mathbf{k}_p^{\perp}\,\boldsymbol{\xi}^{\perp} - \mathbf{k}_i^{\perp}\int_0^t \boldsymbol{\xi}^{\perp}(\tau)\,\mathrm{d}\tau \end{aligned} \tag{3.22}$$

with the reference arc length $\xi_r^{\parallel}$ and the tangential error $e^{\parallel} = \xi^{\parallel} - \xi_r^{\parallel}$. Here, $k_j^{\parallel}$ are appropiate positive constants and $\mathbf{k}_j^{\perp}$ diagonal matrices with appropriate positive entries for all $j \in \{d, p, i\}$. Therein, the integral term was utilised to cope with model uncertainties and disturbances. Similar to the choice of filter constants, the gains for the feedback law (3.22) are chosen to be $k_i = -p_c^3$, $k_p = 3p_c^2$ and $k_d = -3p_c$ for the tangential control, with the negative real controller pole p_c. Analogously, the transversal control gains $k_i\mathbf{I}$, $k_p\mathbf{I}$ and $k_d\mathbf{I}$ were used, where $\mathbf{I}$ denotes the identity matrix of appropriate dimension.

Intuitively, the virtual feedback law (3.22) acts on the tangential and transversal dynamics separately. The origin of the transversal dynamics represents the path, which is stabilised by (3.22), satisfying asymptotic convergence to the path, **O1**. Additionally, the tangential controller makes the output track the tangential reference, thus satisfying **O3**. Hereby it is important to note that the tangential controller can only move the system along the path. Therefore, it cannot make the system deviate from the reference curve and thus the desired invariance property **O2** holds.

Since the system (3.20) is not linear, there is no guarantee for stability. A mathematical proof of stability will be topic of further research. However, in Chapter 4 it is shown in extensive simulation studies that the proposed control law shows stable behaviour in every tested scenario.

4 Simulation and Experimental Validation

To demonstrate the features of the algorithm described in Chapter 3, simulation studies of different systems are presented. The differences between the two proposed control laws are shown for a point mass system in Section 4.1. To illustrate the behaviour of the path feedback controller on systems with nonholonomic constraints, a single-track vehicle model is given in Section 4.2. Furthermore, practical advantages of path following control over trajectory tracking and the distinct benefits of the proposed approach are discussed using a Delta robot (Tripod) in Section 4.3.

Subsequently, the feedback controller based on (3.17) is denoted by Controller 1 or C1. Similarly, the decoupling controller according to (3.18) is referred to as Controller 2, C2.

4.1 Point mass

In the following section, a point mass system is considered to illustrate the differences between the two concepts of the path following controller (3.15). The dynamics of a point mass are given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{x}\\ \dot{\mathbf{x}}\end{bmatrix} = \begin{bmatrix}\dot{\mathbf{x}}\\ \mathbf{0}\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \frac{1}{m}\,\mathbf{I}_{2\times 2}\end{bmatrix}\mathbf{u} \tag{4.1}$$

with the position vector $\mathbf{x} \in \mathbb{R}^2$, the scalar positive mass m, and the input force $\mathbf{u} \in \mathbb{R}^2$. Since a linear controllable system is state feedback linearisable the output $\mathbf{y} = \mathbf{h}(\mathbf{x}) = \mathbf{x}$ is used. Note that despite the fact that the system (4.1) is linear, the closed-loop dynamics are rendered nonlinear by the reference path.

Figure 4.1 shows the output trajectories of the system (4.1) with unit mass, i. e. $m = 1\,\mathrm{kg}$ and an ellipsoid reference path for both controllers. As outer control law for both concepts (3.22) was used with the parameters $k_d^{\parallel} = 3$, $k_p^{\parallel} = 3$, $k_i^{\parallel} = 0$ for the tangential subsystem and $k_d^{\perp} = 10$, $k_p^{\perp} = 25$, $k_i^{\perp} = 0$ for the transversal subsystem. The pole of the reference filter was chosen as $p = -15$. The simulation was run with a sampling time of $T_s = 10\,\mathrm{ms}$ and a maximal tangential velocity $\dot{\xi}_{max}^{\parallel} = 5\,\mathrm{m/s}$. Since the dynamics of the system degenerate to (3.20) along the path using either C1 or C2, differences occur predominantly in significant distance to the path. Therefore, the initial conditions are chosen with the output far off the reference path. Starting- and endpoint are marked using a circle and a cross, respectively. Both control methods quickly converge to the path, but C1 first goes faster to the reference path and then achieves the desired tangential movement.

The arc length, the tangential state, and its time derivative for the simulation are depicted in Figure 4.2. Here, the tangential velocity clearly shows that the decoupling controller C2 reaches the reference velocity quicker than C1. Both controllers closely track the tangential reference.

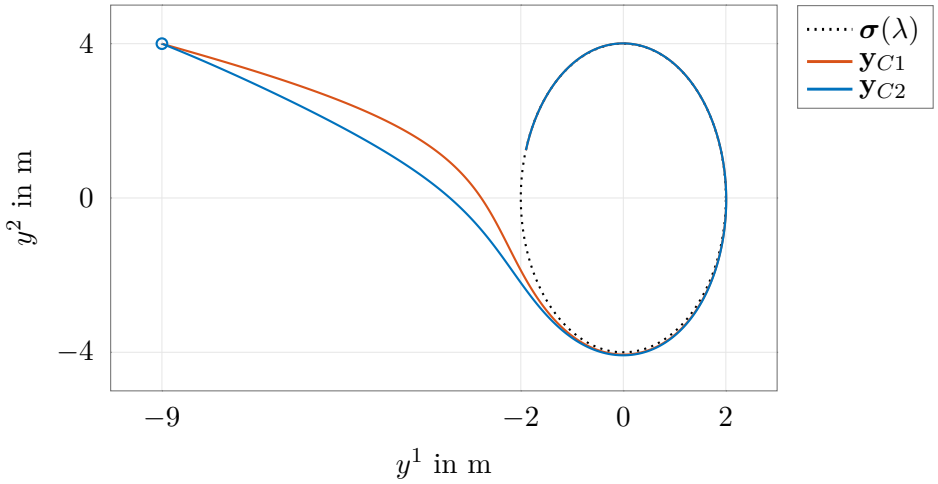

Figure 4.1: Path of the output controlled by the pseudo-inverse (C1) and decoupling (C2) controllers.

Recall, that the pseudo-inverse controller C1 solves the feedback linearisation problem optimal in the sense of the Euclidean norm. Because the norm of the virtual control input in transversal direction is larger than the one in tangential direction, the reduction of the distance to the path is prioritised compared to C2. Therefore, C1 is well suited for applications, where a balance between tangential motion and distance to the path is beneficial. The controller C2 can be used to decouple tangential and transversal goals.

4.2 Single-Track Vehicle

A path following algorithm can be benchmarked using a single-track vehicle in two ways. First, a car has restrictive limits on the steering angle and secondly the system has nonholonomic constraints. A single-track model as depicted in Figure 4.3 can be regarded as a simple model of a car. Although the vertical and transversal as well as the pitch and roll dynamics of the car are neglected, the single-track model includes the nonholonomic tire-ground contact and the steering dynamics.

4.2.1 Mathematical Model

Using only kinematic relations and the nonholonomic constraint of the front (steered) wheels, the dynamics can be described by

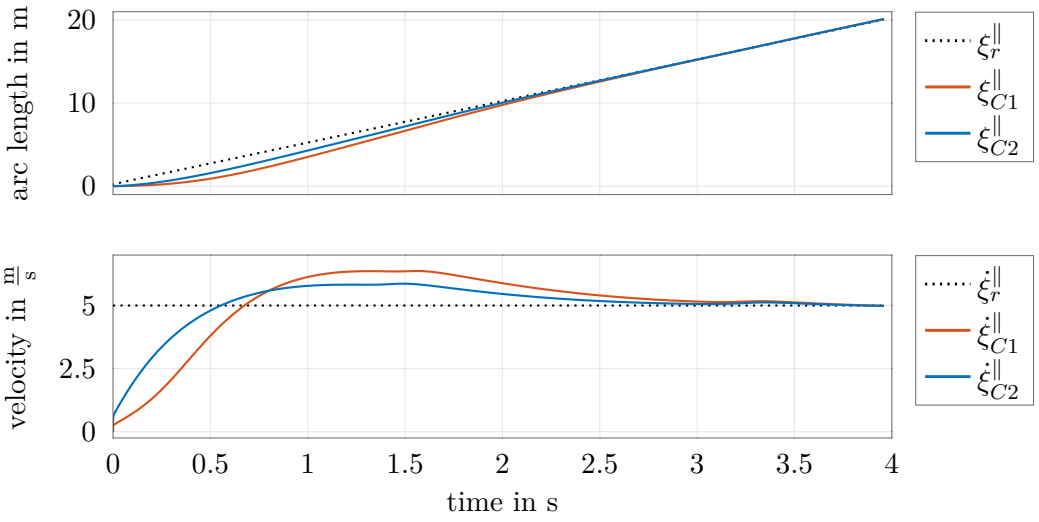

Figure 4.2: Comparison of the arc length and its time derivative for the point mass system using C1 and C2.

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} y^1 \\ y^2 \\ \psi \\ v \end{bmatrix} = \begin{bmatrix} v\cos(\psi) \\ v\sin(\psi) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{v}{l}\tan(\delta) \\ a \end{bmatrix}, \tag{4.2}$$

with the position of the car (y^1, y^2), the heading angle ψ and the velocity v. The wheelbase, the distance between the front and rear axle, is denoted by l. The acceleration a and the tangent of the steering angle $\tan(\delta)$ constitute the inputs. Note that the acceleration input can be substituted by the driving force F with $a = F/m$, where m is the mass of the vehicle. The output consists of the position of the centre of the back tire

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) = \begin{bmatrix} y^1 \\ y^2 \end{bmatrix}.$$

The Lie derivatives along the system dynamics $\mathbf{f}(\mathbf{x})$ and the input vector field $\mathbf{g}(\mathbf{x})$ can be found to be

$$\mathrm{L}_{\mathbf{f}}\mathbf{h} = \begin{bmatrix} v\cos(\psi) \\ v\sin(\psi) \end{bmatrix}$$

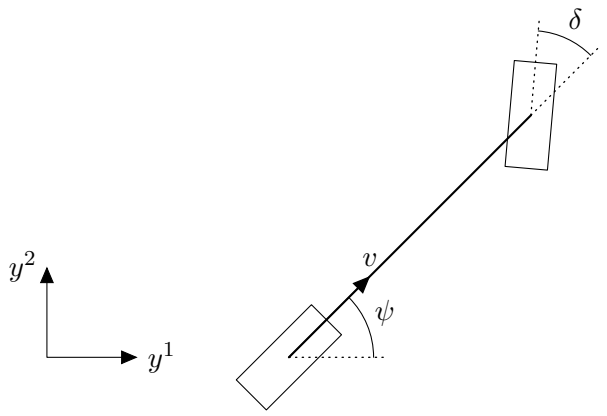

Figure 4.3: Illustration of the single-track vehicle model.

and $\mathrm{L}_{\mathbf{g}}\mathbf{h} = \mathbf{0}$, respectively. The derivatives of second order are $\mathrm{L}_{\mathbf{f}}^{2}\mathbf{h} = \mathbf{0}$ and

$$\mathbf{D} = \mathrm{L}_{\mathbf{g}}\mathrm{L}_{\mathbf{f}}\mathbf{h} = \begin{bmatrix} -\dfrac{v^2}{l}\sin(\psi) & \cos(\psi) \\ \dfrac{v^2}{l}\cos(\psi) & \sin(\psi) \end{bmatrix} . \tag{4.3}$$

Since $\mathrm{L}_{\mathbf{g}}\mathbf{h} = \mathbf{0}$ and (4.3) has full rank at any point in the state space where $v \neq 0$, the model has a locally well defined relative degree of $\{2, 2\}$ which satisfies Assumption 2. Because of the fact that the relative degree matches the system order, the system is exact feedback linearisable. The inverse of the decoupling matrix reads as

$$\mathbf{D}^{-1} = \begin{bmatrix} -\dfrac{l}{v^2}\sin(\psi) & \dfrac{l}{v^2}\cos(\psi) \\ \cos(\psi) & \sin(\psi) \end{bmatrix} .$$

These terms substituted into (3.17) and (3.18) constitute the feedback C1 and C2, respectively. For further details see Appendix A.

4.2.2 Simulation Results

In this section, a simulation is presented where the system (4.2) is controlled using the pseudo-inverse controller C1 to follow an ellipsoid reference. Because the controllers are equivalent in a close neighbourhood of the path, all observations for C1 hold true for C2. However, due to the coupling of tangential and transversal subsystem, the convergence to the path is prioritised using C1, see Section 4.1. This coupling leads to an undesired overshoot for the nonholonomic system.

The wheelbase is $l = 1\,\mathrm{m}$ and the mass of the vehicle $m = 1\,\mathrm{kg}$. The reference path is

given by

$$\boldsymbol{\sigma}(\lambda) = \begin{bmatrix} a\cos(\lambda) \\ b\sin(\lambda) \end{bmatrix} ,$$

with the major and minor semi-axes $a = 5$, $b = 3$, respectively. The steering angle was limited to $\delta_{max} = \pm 35°$ and the input constraint on the acceleration was set to $a_{max} = \pm 1\,\mathrm{m/s^2}$. The initial conditions were chosen as $\mathbf{x}_0 = (1, -3, 0, 0.1)$, where the heading angle is measured relative to the y^1-axis. Here, the outer controller (3.22) was used with the parameters $k_d^{\parallel} = 15$, $k_p^{\parallel} = 75$, $k_i^{\parallel} = 0$ for the tangential subsystem and $k_d^{\perp} = 18$, $k_p^{\perp} = 81$, $k_i^{\perp} = 0$ for the transversal subsystem. The pole of the reference filter (3.21) was set to $p = -1$ and the maximum tangential velocity $\xi_{max}^{\parallel} = 5\,\mathrm{m/s}$, see Section 3.3.5. The sampling time is set to $T_s = 1\,\mathrm{ms}$.

Figure 4.4 depicts an output path for this scenario. The model quickly converges to

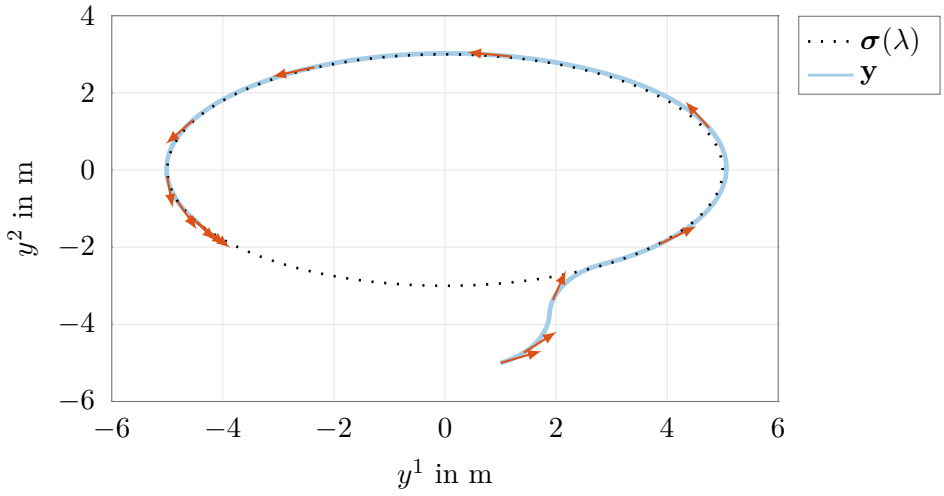

Figure 4.4: Output path of the single-track vehicle controlled to follow an elliptical reference curve. Arrows indicate the heading direction in 1 s intervals.

the reference path. However, due to the input constraints and the nonholonomic system constraint the output overshoots the path. This effect can also be observed in Figure 4.5, where the virtual states are depicted together with the system input. Here, the input saturation limits are depicted as dashed lines. After starting with fully saturated inputs, the steering is reversed at about two seconds. Therefore, the car is not tangential to the ellipse and the tangential velocity profile cannot be exactly tracked. The transversal error to the path is quickly reduced, but the saturated steering causes the car to overshoot the reference at around three seconds.

After following the reference closely, the constraint on the maximum deceleration is reached at nine seconds. Note that even though the reference cannot be tracked, there is no deviation from the reference path. This is a consequence of the invariance property **O2**. However, the invariance property holds only true for the unconstrained system, since

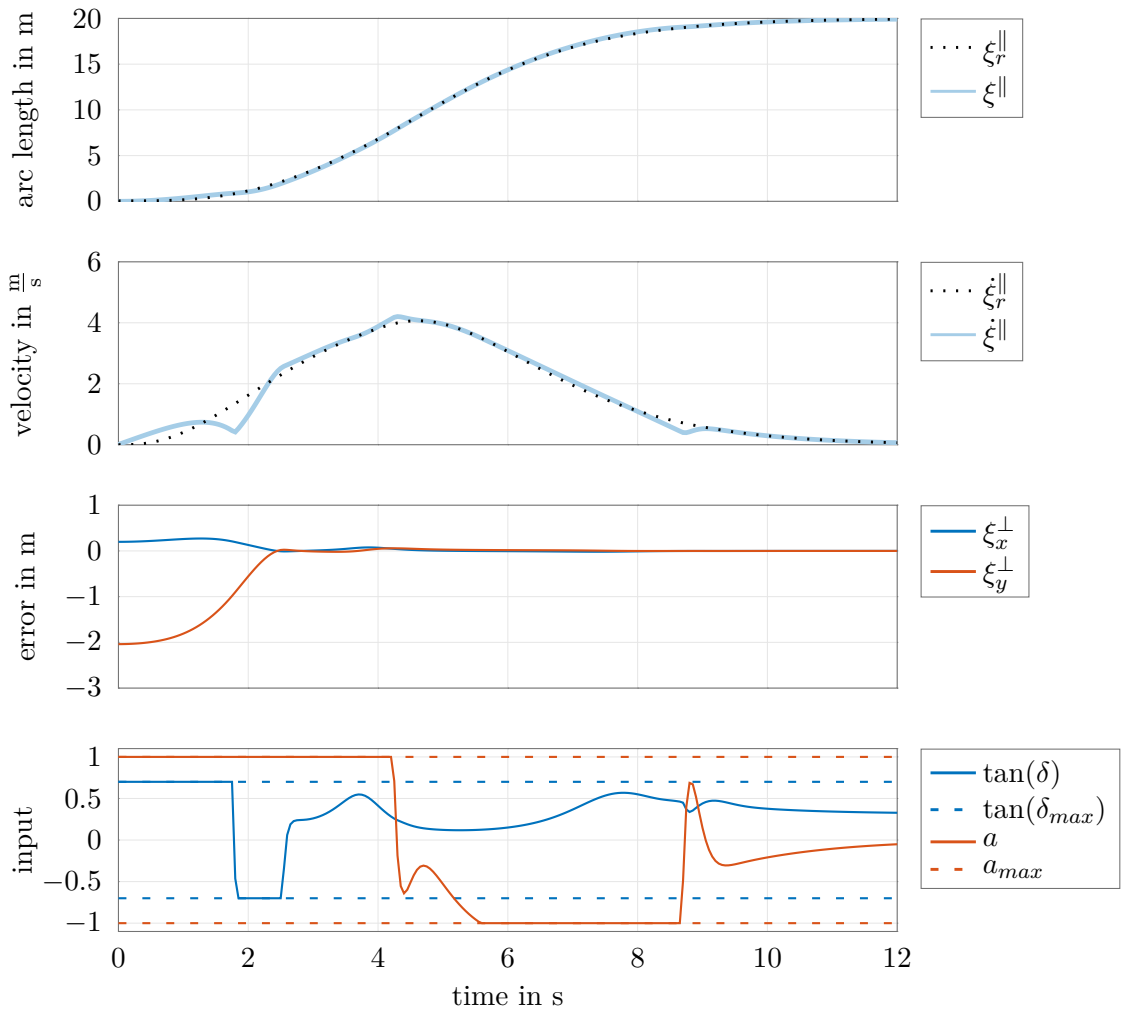

Figure 4.5: Simulation results for the single-track vehicle following an elliptical path.

the static feedback law cannot systematically deal with constraints.

4.3 Tripod

In the following, the application of the proposed algorithm to a Delta robot with linear drives is presented, see Figure 4.6. First, a mathematical model is introduced followed by simulated and experimental results of the proposed algorithm. The Delta robot with rotary drives was introduced by Clavel [33] and was comprehensively treated in literature, see, e. g., [34].

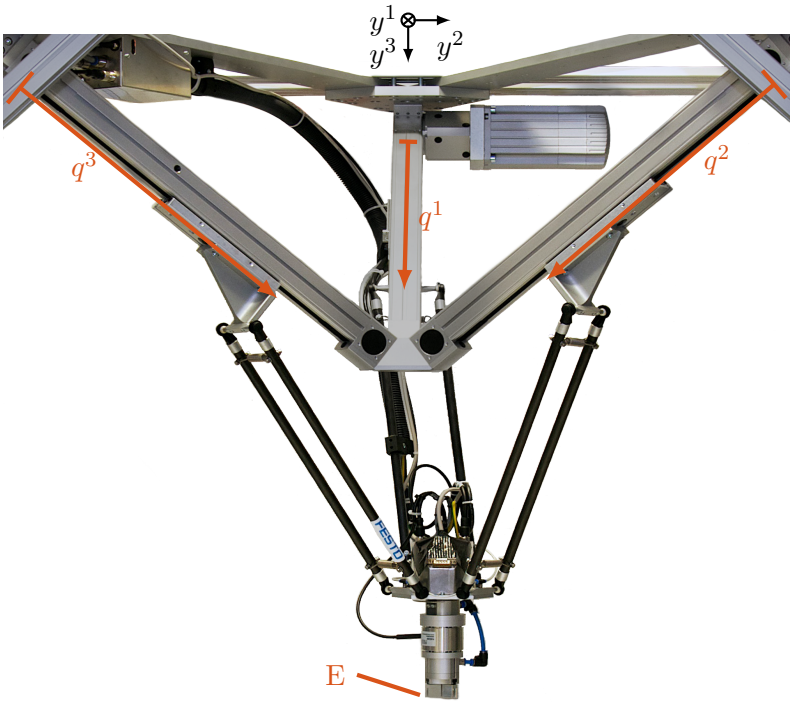

Figure 4.6: FESTO EXPT-45 Delta robot.

4.3.1 Mathematical Model

The Delta robot has three translational degrees of freedom which can be described by the coordinates of the end-effector E and their time derivative $\mathbf{y} = (\overrightarrow{OE})$ and $\dot{\mathbf{y}}$, respectively. Likewise, the position and velocity of the sleds $\mathbf{q}$ and $\dot{\mathbf{q}}$, respectively, can be used.

In the model used for control design, all masses are considered as locally concentrated point masses. The mass of the parallelogram-rods is equally distributed between the adjacent sled and the end-effector. Furthermore, the friction in the motors, sleds and joints is neglected.

The dynamics describing the Delta robot can be represented in the input-affine state space form

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} = \begin{bmatrix}\dot{\mathbf{q}}\\ -\mathbf{M}^{-1}\left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{k}(\mathbf{q})\right)\end{bmatrix} + \begin{bmatrix}0\\ \mathbf{M}^{-1}\end{bmatrix}\boldsymbol{\tau} \tag{4.4}$$

with the input force on the sleds $\boldsymbol{\tau}$ and the state consisting of sled positions and velocities $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^{\mathrm{T}}$, as well as the positive definite mass matrix $\mathbf{M}$, the Coriolis matrix $\mathbf{C}$, and the potential terms $\mathbf{k}(\mathbf{q})$. The output equation is given by the forward kinematics

$$\mathbf{y} = \mathbf{h}(\mathbf{q}) \ , \tag{4.5}$$

which describes the end-effector position as a function of the sled positions. Using these equations, the decoupling matrix $\mathbf{D}$ and its inverse can be calculated. The system (4.4) is

augmented by the projected path parameter state λ^* according to (3.12). Furthermore, the augmented output $\mathbf{y}_A$ is given by (3.13).

Using these terms the control algorithm proceeds as follows:

1. The current path parameter value is obtained by solving (3.9) using the Newton algorithm and the adaptive step size algorithm presented in [35].
2. By using the kinematic relations and measurements of $\mathbf{q}$, the tangential and orthogonal states $\boldsymbol{\xi}$ are calculated.
3. The virtual tangential and transversal control inputs are calculated using (3.22).
4. The feedback transformation (3.18) is used to calculate the corresponding forces applied to the actuators.

4.3.2 Simulation Results

To validate the approaches of this thesis, a path calculated using the methods of Section 2.4.1 was followed using the path following controller of Section 3.3.4 in simulation. To this end, the mathematical model of the Delta robot was used as test system. The simulation was carried out on a more detailed model including friction. Furthermore, inaccurate sensor measurements for the velocities were modelled by white noise.

The virtual control inputs for the pseudo-inverse control law C1 are chosen according to (3.22). Here, the feedback gains are obtained by placing the poles of the tangential system at $p_i^{\parallel} = -35$, $i = 1, 2, 3$ and the poles of the transversal system at $p_i^{\perp} = -50$, $i = 1, 2, 3$. The tangential reference position, velocity and acceleration are obtained using the reference filter described in Section 3.3.5. The maximal velocity is set to $\dot{\xi}_{max}^{\parallel} = 0.5\,\mathrm{m/s}$ and the poles of the reference state-filter are placed at $p_i^{ref} = -30$, $i = 1, 2, 3$. The sample time of the controller is set to $T_s = 1\,\mathrm{ms}$.

The workspace of the robot contains three obstacles, which have to be avoided. This problem can be compared to a pick and place action performed by the Delta robot. The cost function in the optimal RRT ensures that the path is as short as possible. Thus, the required effort to position the end-effector of the robot is kept low. A three-dimensional illustration of the workspace, the end-effector path and the reference curve are depicted in Figure 4.7. The corresponding time dependent states and control inputs as well as the errors are depicted in Figure 4.8. Here, it is shown that even for a complex mechanical system with friction and measurement noise, the controller tracks the output precisely. The simulated measurement noise causes a noisy control actuation. Note that the reference path is calculated using a third order spline interpolation as described in Section 2.4.2. Thus, the path is twice continuously differentiable, while the controller requires at least three continuous derivatives for the control law to be continuous, see Assumption 4. However, the achieved performance shows that the introduced noise is tolerable in practice. The norm of the deviation from the reference path stays below 1 mm during the whole manoeuvre. Despite the friction included in the simulation model and the measurement noise, the controller follows the reference with high precision.

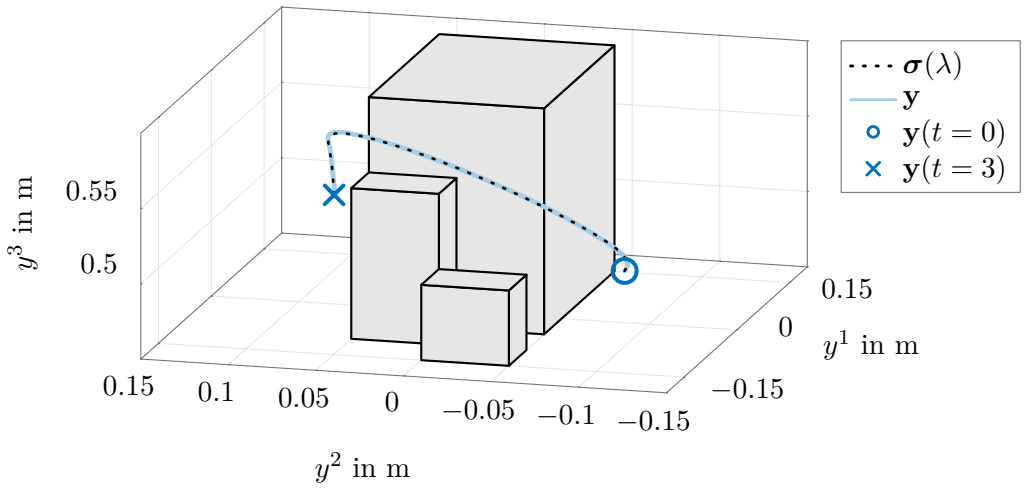

Figure 4.7: Output path of the Tripod end-effector together with the spline interpolated RRT reference curve.

4.3.3 Experimental Validation

The path following approach derived in Section 3.3.4 is experimentally verified on the Delta Robot depicted in Figure 4.6. This parallel kinematic robot FESTO EXPT-45 is actuated by three linear drives. The linear actuator distances are measured using optical encoders. The currents applied of the motors serve as control inputs. The output is obtained by using the measurements of the linear actuator positions and the forward kinematic equations of the robot $\mathbf{y} = \mathbf{h}(\mathbf{q})$. Similarly, the velocity is obtained numerically by applying finite differences. The control law is iterated with a sampling time of $T_s = 1\,\text{ms}$ on a MATLAB D-Space system.

The virtual control inputs are chosen according to (3.22), where the feedback gains are obtained by placing the poles of the tangential system at $p_i^{\parallel} = -25$, $i = 1, 2, 3$ and the poles of the transversal system at $p_i^{\perp} = -35$, $i = 1, 2, 3$. The tangential reference position, velocity and acceleration are obtained using the reference filter described in Section 3.3.5. The maximal velocity is set to $\dot{\xi}_{max}^{\parallel} = 0.5\,\text{m/s}$ and the poles of the reference state-filter are placed at $p_i^{ref} = -50$, $i = 1, 2, 3$.

Figure 4.9 shows the trajectory of the end-effector following a curved lemniscate reference. The results shown are for the pseudo-inverse control law C1. Since the differences between the controllers C1 and C2 are negligible in the close neighbourhood to the path and big deviations are not realisable due to workspace constraints, the results for C2 are omitted.

Figure 4.10 shows the corresponding tangential and transversal states, as well as the input signal. The reference point along the arc length $\xi_r^{\parallel}$ is followed with high precision. Even though the numerical approximation of the tangential velocity introduces considerable noise to the control signal, the tangential reference velocity is followed closely. The distance to the path $\|\xi^{\perp}\|$ stays well below 1 mm. The static offset in actuation is used to counteract gravity.

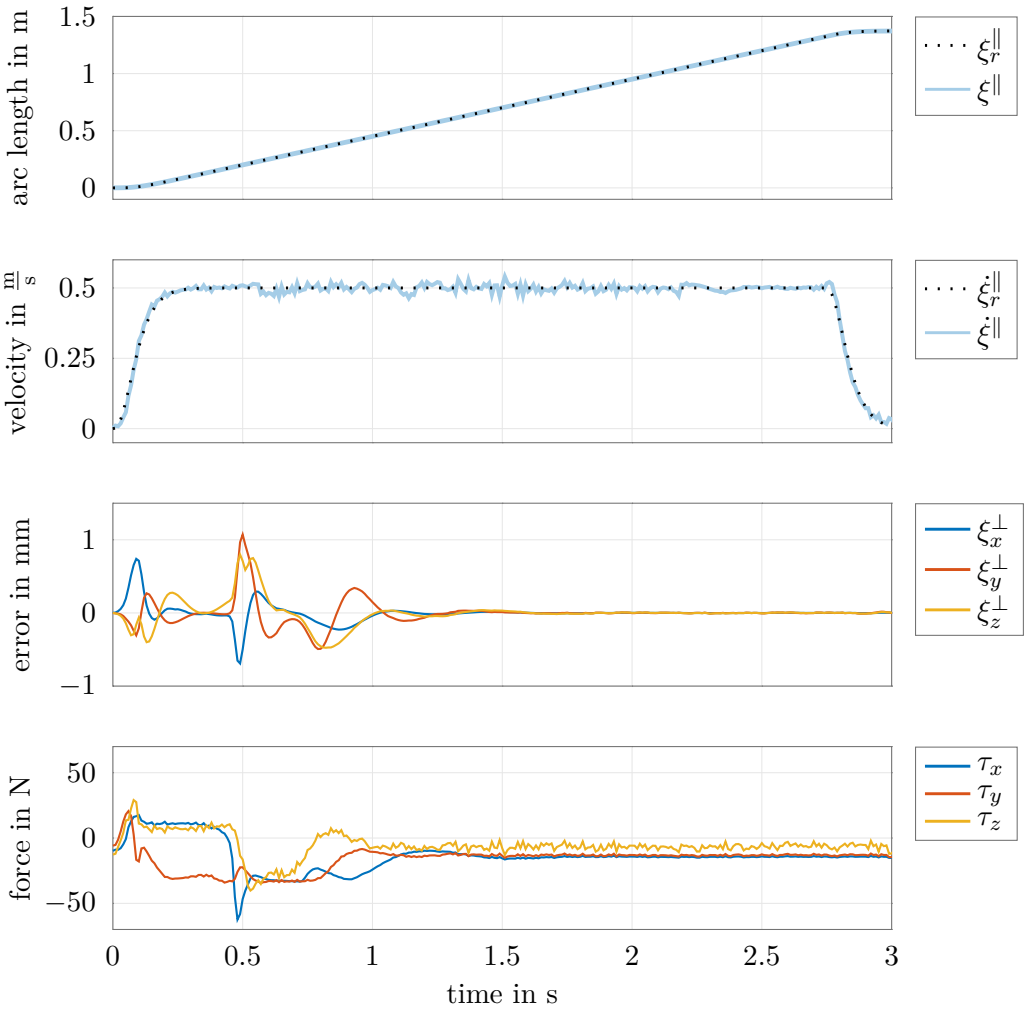

Figure 4.8: Simulation results of the Tripod controlled along the RRT interpolated spline.

The simulation results obtained in Section 4.3.2 compare very well to these experimental results. While the curved lemniscate used in the experiment requires ongoing correctional inputs, the simulated RRT path can be followed using less input effort. However, the experiment shows an even more precise tracking of the reference curve. Furthermore, the noise level of the simulation closely matches the measurement noise of the practical system.

The presented results can be reproduced by the controller C2. Despite the lack of a stability proof, both controllers C1 and C2 show an excellent path-following behaviour.

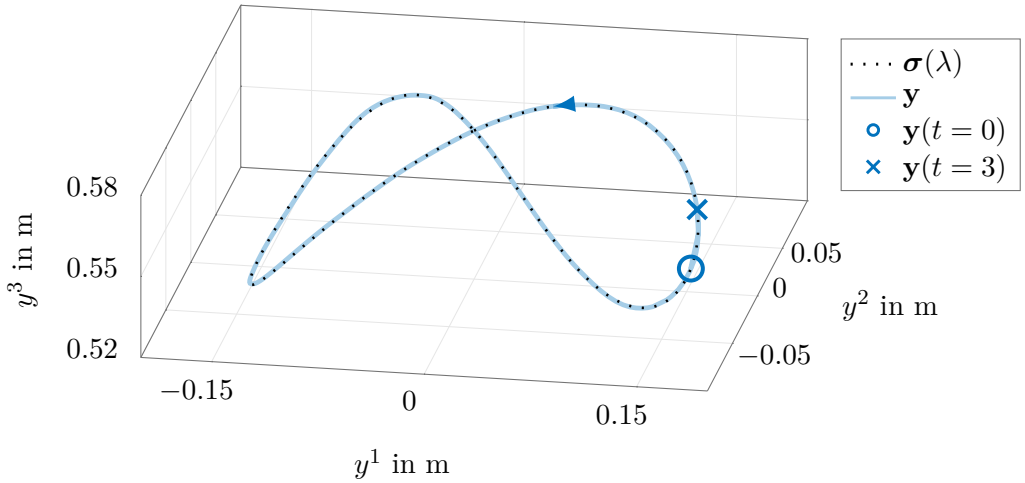

Figure 4.9: Output path of the Tripod end-effector along a curved lemniscate reference curve.

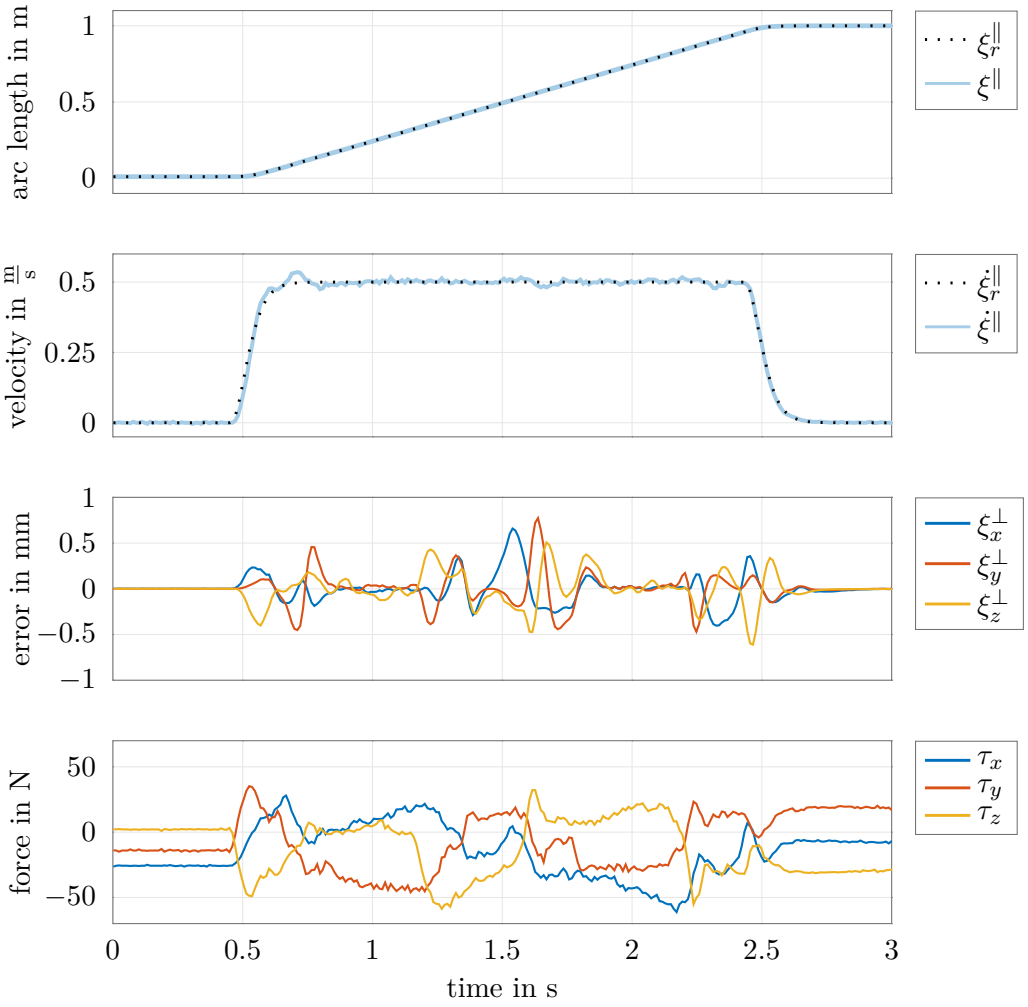

Figure 4.10: Experimental results of the Tripod controlled along the curved lemniscate path.

5 Conclusion

In this thesis, a systematic way of planning and following a geometric reference curve for mechanical systems was presented. Therefore, this task was separated into a planning and a control problem.

The planning problem was addressed in Chapter 2 by introducing the Rapidly exploring Random Tree algorithm and an extension, the optimal RRT, to solve the discrete path planning problem. These algorithms iteratively find collision free connections between randomly sampled states. Because the algorithm iteratively converges to the optimal solution it can be cancelled at any time while still providing a feasible solution. By Monte-Carlo simulations the rate of convergence of this stochastic algorithm was checked. Because of its boolean collision check, the RRT is well suited for highly cluttered environments.

The global optimality of the optimal RRT comes at the cost of computation time. To increase the efficiency of the algorithm, different approaches have been proposed in literature. The application of a branch and bound technique to the graph or the restriction of the RRT to a local method are only two examples. In this work a simplified dynamic model was applied to reduce the computational effort. The RRT was used in this thesis to plan a path in the three-dimensional space, rather than in the state space of the system. The consequences of this choice are twofold: while the computational efficiency is significantly increased, strict collision avoidance cannot be guaranteed.

The interpolated path found by the RRT was used as a geometric reference curve for the path following controller proposed in Chapter 3. This method, which is based on exact feedback linearisation, is applicable to systems with a well defined relative degree, such as mechanical systems studied in this thesis.

To this end the differential geometric method of exact feedback linearisation for systems with full relative degree was used on a novel coordinate frame. This method is used to counteract nonlinearities and simplify the dynamics of a system such that in new coordinates they are linear. The novelty of the proposed approach is that a static Cartesian coordinate system is used for the distance to the path. This choice establishes an intuitive notion of the used states and the path following error. To use static coordinates and avoid singularities for the transversal dynamics a redundant coordinate is introduced. Two variations of the algorithm are proposed, one of which minimises the quadratic error while the other aims at decoupling the tangential and transversal dynamics.

Because the coordinates are not moving with the output the system cannot be rendered linear. Therefore a stability proof cannot be found. However, simulations of a variety of systems demonstrate the feasibility of the control scheme. Experimental results on a tripod robot further underline the practical applicability of the algorithm developed in this thesis. The method proved to provide a high degree of precision even when tracking paths with high curvature.

A Lie Derivatives of the Augmented System

In this appendix, the time derivatives of the output in Equation (3.13) are derived. The time derivatives of the augmented system output are given in terms of the Lie derivatives as

$$\dot{\mathbf{y}}_A = \mathrm{L}_{\mathbf{f}_A}\mathbf{h}_A + \underbrace{\mathrm{L}_{\mathbf{g}_A}\mathbf{h}_A}_{=\mathbf{0}}\,\mathbf{u} = \begin{bmatrix} \dot{\xi}^{\parallel} \\ \dot{\boldsymbol{\xi}}^{\perp} \end{bmatrix}$$

and

$$\ddot{\mathbf{y}}_A = \mathbf{E}_A(\mathbf{x}_A) + \mathbf{D}_A(\mathbf{x}_A)\,\mathbf{u} = \begin{bmatrix} \mathrm{L}^2_{\mathbf{f}_A}\xi^{\parallel} \\ \mathrm{L}^2_{\mathbf{f}_A}\boldsymbol{\xi}^{\perp} \end{bmatrix} + \begin{bmatrix} \mathrm{L}_{\mathbf{g}_A}\mathrm{L}_{\mathbf{f}_A}\xi^{\parallel} \\ \mathrm{L}_{\mathbf{g}_A}\mathrm{L}_{\mathbf{f}_A}\boldsymbol{\xi}^{\perp} \end{bmatrix}\,\mathbf{u}\ .$$

To avoid complex formulations the Lie derivatives are calculated by first taking the time derivatives and factoring terms including the input.

For the tangential output the velocity along the tangent of the path is required. It can be found by using the Leibniz integration rule to be

$$\dot{\xi}^{\parallel} = \|\boldsymbol{\sigma}'(\lambda^*)\|\ \dot{\lambda}^* - \underbrace{\frac{\mathrm{d}\lambda_0}{\mathrm{d}t}}_{=0}\|\boldsymbol{\sigma}'(\lambda_0)\| + \int_{\lambda_0}^{\lambda^*} \underbrace{\left(\frac{\partial\|\boldsymbol{\sigma}'(\zeta)\|}{\partial t}\right)}_{=0}\mathrm{d}\zeta\ .$$

Note that neither the initial path parameter $\lambda_0$nor the norm of the tangent direction depend on time. In the following, all dependencies are omitted for clearness of exposition. The second time derivative of the arc length is

$$\ddot{\xi}^{\parallel} = \frac{(\boldsymbol{\sigma}')^{\mathrm{T}}}{\|\boldsymbol{\sigma}'\|}\ \boldsymbol{\sigma}''\ (\dot{\lambda}^*)^2 + \|\boldsymbol{\sigma}'\|\ \ddot{\lambda}^*\ ,$$

where all expressions are evaluated at λ^*. Recall that λ^* is the path parameter corresponding to the system output projected onto the path. Therefore, it is actuated by the system and consequently has the same relative degree. This observation is used to note that the first derivatives are not directly affected by the input and this yields

$$\mathrm{L}^2_{\mathbf{f}_A}\xi^{\parallel} = \frac{(\boldsymbol{\sigma}')^{\mathrm{T}}}{\|\boldsymbol{\sigma}'\|}\ \boldsymbol{\sigma}''\ (\dot{\lambda}^*)^2 + \|\boldsymbol{\sigma}'\|\ \mathrm{L}^2_{\mathbf{f}_A}\lambda^*$$

$$\mathrm{L}_{\mathbf{g}_A}\mathrm{L}_{\mathbf{f}_A}\xi^{\parallel} = \|\boldsymbol{\sigma}'\|\ \mathrm{L}_{\mathbf{g}_A}\mathrm{L}_{\mathbf{f}_A}\lambda^*\ .$$

Using

$$\ddot{\lambda}^* = \frac{\mathrm{d}}{\mathrm{d}t}\left[\kappa\,(\boldsymbol{\sigma}')^{\mathrm{T}}\,\dot{\mathbf{y}}\right] = \dot{\kappa}\,(\boldsymbol{\sigma}')^{\mathrm{T}}\,\dot{\mathbf{y}}\ + \kappa\,(\boldsymbol{\sigma}'')^{\mathrm{T}}\,\dot{\lambda}^*\,\dot{\mathbf{y}}\ + \kappa\,(\boldsymbol{\sigma}')^{\mathrm{T}}\,\ddot{\mathbf{y}}$$

and again exploiting the relative degree of the system (3.12), the only term including the input is $\ddot{\mathbf{y}}$. From this the Lie derivatives are deduced as

$$\begin{aligned} \mathrm{L}^2_{\mathbf{f}_A}\lambda^* &= \dot{\kappa}\,(\boldsymbol{\sigma}')^{\mathrm{T}}\,\mathrm{L}_{\mathbf{f}}\mathbf{h} \;+\kappa\,(\boldsymbol{\sigma}'')^{\mathrm{T}}\,\dot{\lambda}^*\,\mathrm{L}_{\mathbf{f}}\mathbf{h} \;+\kappa\,(\boldsymbol{\sigma}')^{\mathrm{T}}\,\mathrm{L}^2_{\mathbf{f}}\mathbf{h} \\ \mathrm{L}_{\mathbf{g}_A}\mathrm{L}_{\mathbf{f}_A}\lambda^* &= \kappa\,(\boldsymbol{\sigma}')^{\mathrm{T}}\,\mathrm{L}_{\mathbf{g}}\mathrm{L}_{\mathbf{f}}\mathbf{h}\;. \end{aligned}$$

The time derivative of κ reads as

$$\begin{aligned} \dot{\kappa} &= \kappa^2\,\frac{\mathrm{d}}{\mathrm{d}t}\left[(\mathbf{y}-\boldsymbol{\sigma})^{\mathrm{T}}\,\boldsymbol{\sigma}'' \;-(\boldsymbol{\sigma}')^{\mathrm{T}}\,\boldsymbol{\sigma}'\right] \\ &= \kappa^2\left[\left(\dot{\mathbf{y}}-3\boldsymbol{\sigma}'\,\dot{\lambda}^*\right)^{\mathrm{T}}\,\boldsymbol{\sigma}'' \;+(\mathbf{y}-\boldsymbol{\sigma})^{\mathrm{T}}\;\boldsymbol{\sigma}'''\,\dot{\lambda}^*\right]\;. \end{aligned}$$

By the definition of the Lie derivative operator, it is additive with respect to its second argument. Thus, the second order Lie derivatives of the transversal state can be expressed as

$$\begin{aligned} \mathrm{L}^2_{\mathbf{f}_A}\boldsymbol{\xi}^{\perp} &= \mathrm{L}^2_{\mathbf{f}}\mathbf{h} - \;\mathrm{L}^2_{\mathbf{f}_A}\boldsymbol{\sigma} \\ \mathrm{L}_{\mathbf{g}_A}\mathrm{L}_{\mathbf{f}_A}\boldsymbol{\xi}^{\perp} &= \mathrm{L}_{\mathbf{g}}\mathrm{L}_{\mathbf{f}}\mathbf{h} - \mathrm{L}_{\mathbf{g}_A}\mathrm{L}_{\mathbf{f}_A}\boldsymbol{\sigma}\;, \end{aligned}$$

with a term depending on the original system output and a term depending on the path. Because the path $\boldsymbol{\sigma}$ depends only on the path parameter a simple calculation yields

$$\mathrm{L}_{\mathbf{f}_A}\boldsymbol{\sigma} = \boldsymbol{\sigma}'\;\dot{\lambda}^*$$

and

$$\begin{aligned} \mathrm{L}^2_{\mathbf{f}_A}\boldsymbol{\sigma} &= \boldsymbol{\sigma}''\;(\dot{\lambda}^*)^2 + \boldsymbol{\sigma}'\;\mathrm{L}^2_{\mathbf{f}_A}\lambda^* \\ \mathrm{L}_{\mathbf{g}_A}\mathrm{L}_{\mathbf{f}_A}\boldsymbol{\sigma} &= \boldsymbol{\sigma}'\;\mathrm{L}_{\mathbf{g}_A}\mathrm{L}_{\mathbf{f}_A}\lambda^*\;. \end{aligned}$$

Thus, the second Lie derivative of the orthogonal states reads as

$$\mathrm{L}^2_{\mathbf{f}_A}\boldsymbol{\xi}^{\perp} = \mathrm{L}^2_{\mathbf{f}}\mathbf{h} - \;\boldsymbol{\sigma}''\;(\dot{\lambda}^*)^2 - \boldsymbol{\sigma}'\;\mathrm{L}^2_{\mathbf{f}_A}\lambda^*\;.$$

B Properties of Projections

The projection onto a line is given by

$$\mathbf{P}_{line} = \mathbf{v}\mathbf{v}^{\mathrm{T}} ,$$

where $\mathbf{v}$ is a vector of unit length in the direction of the line. The kernel and range of $\mathbf{P}_{line}$ and

$$\mathbf{P} = \mathbf{I} - \mathbf{P}_{line}$$

are complementary, thus $\mathbf{P}$ is a projection onto the orthogonal subspace of the line.

A projection matrix $\mathbf{P}$ has two properties which are important in the context of this thesis:

- A projection is idempotent, i. e. $\mathbf{P}\mathbf{P} = \mathbf{P}$.
- The singular values of $\mathbf{P}$ are either 1 or 0.
- $\mathbf{P}$ is self adjoint, i. e. since it is real valued it is symmetric.

It is important to note that not only $\mathbf{P}\mathbf{v} = \mathbf{0}$, but also because $\mathbf{P}$ is symmetric $\mathbf{v}^{\mathrm{T}}\mathbf{P} = \mathbf{0}$.

The matrix

$$\mathbf{P}_{\kappa} = \mathbf{I} - \kappa\ \mathbf{v}\mathbf{v}^{\mathrm{T}}$$

is a projection if and only if $\kappa = \|\mathbf{v}\|^{-2}$. In the general case, $\mathbf{P}_{\kappa}$ has full rank and is no projection. However, it still holds that $\mathbf{P}_{\kappa}\ \mathbf{P} = \mathbf{P}$ since

$$\begin{aligned}
\mathbf{P}_{\kappa}\ \mathbf{P} &= \left(\mathbf{I} - \kappa\mathbf{v}\mathbf{v}^{\mathrm{T}}\right)\ \left(\mathbf{I} - \frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\|\mathbf{v}\|^2}\right) \\
&= \mathbf{I} - \frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\|\mathbf{v}\|^2} - \kappa\mathbf{v}\mathbf{v}^{\mathrm{T}} + \kappa\frac{\mathbf{v}\ \mathbf{v}^{\mathrm{T}}\mathbf{v}\ \mathbf{v}^{\mathrm{T}}}{\mathbf{v}^{\mathrm{T}}\mathbf{v}} \\
&= \mathbf{I} - \frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\|\mathbf{v}\|^2}\ =\ \mathbf{P}\ .
\end{aligned}$$

A thorough treatment of projections can be found in, e. g., [36].

Bibliography

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[2] J. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[3] R. N. Jazar, *Theory of Applied Robotics*, 1st. New York: Springer, 2007.

[4] J.-P. Laumond, *Robot Motion Planning and Control*. Berlin, Germany: Springer, 1998.

[5] T. Lozano-Pérez, "Spatial planning: a configuration space approach," *IEEE Transactions on Computers*, vol. C-32, no. 2, pp. 108–120, 1983.

[6] J. T. Schwartz and M. Sharir, "On the "piano movers" problem i. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Communications on Pure and Applied Mathematics*, vol. 36, no. 3, pp. 345–398, 1983.

[7] J. Canny, "The complexity of robot motion planning," PhD thesis, Massachusetts Institute of Technology, Boston, USA, 1988.

[8] R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 2, pp. 224–233, 1985.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[10] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of the IEEE Conference on Robotics and Automation*, vol. 2, Sacramento, USA, 1991, pp. 1398–1404.

[11] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, no. 3, pp. 207–222,

[12] J. Barraquand and J.-C. Latombe, "Nonholonomic multibody mobile robots: controllability and motion planning in the presence of obstacles," *Algorithmica*, vol. 10, no. 2, pp. 121–155, 1993.

[13] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[14] E. Frazzoli, M. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 25, pp. 116–129, 2000.

[15] S. Karaman, "Sampling-based algorithms for optimal path planning problems," PhD thesis, Massachusetts Institute of Technology, Boston, USA, 2012.

[16] S. M. LaValle, "Rapidly-exploring random trees: a new tool for path planning," Computer Science Dept., Iowa State University, Technical Report 1998 Vol. 11, 1998.

[17] P. Bertsekas Dimitri, *Dynamic Programming and Optimal Control: Approximate dynamic programming*. Nashua, USA: Athena Scientific, 2012.

[18] S. D. Conte and C. W. D. Boor, *Elementary Numerical Analysis: An Algorithmic Approach*, 3rd. New York, USA: McGraw-Hill Higher Education, 1980.

[19] A. Banaszuk and J. Hauser, "Feedback linearization of transverse dynamics for periodic orbits," in *Proceedings of the IEEE Decision and Control Conference*, vol. 2, Lake Buena Vista, USA, 1994, pp. 1639–1644.

[20] C. Nielsen and M. Maggiore, "Transverse feedback linearization of multi-input systems," in *Proceedings of the IEEE European Control Conference on Decision and Control*, Seville, Spain, 2005, pp. 4897–4902.

[21] C. Nielsen and M. Maggiore, "Further results on transverse feedback linearization of multi-input systems," in *Proceedings of the IEEE Conference on Decision and Control*, San Diego, USA, 2006, pp. 3819–3824.

[22] A. Hladio, C. Nielsen, and D. Wang, "Path following for a class of mechanical systems," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2380–2390, 2013.

[23] M. Böck and A. Kugi, "Manifold stabilization and path-following control for flat systems with application to a laboratory tower crane," in *Proceedings of the IEEE Decision and Control Conference*, Los Angeles, USA, 2014, pp. 4529–4535.

[24] T. Faulwasser, *Optimization-based solutions to constrained trajectory-tracking and path-following problems*, ser. Contributions in Systems Theory and Automatic Control. Aachen, Germany: Shaker, 2013.

[25] L. Consolini, M. Maggiore, C. Nielsen, and M. Tosques, "Path following for the PVTOL aircraft," *Automatica*, vol. 46, no. 8, pp. 1284–1296, 2010.

[26] R. Gill, D. Kulić, and C. Nielsen, "Spline path following for redundant mechanical systems," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1378–1392, Dec. 2015.

[27] B. Bischof, T. Glück, and A. Kugi, "Combined path following and compliance control for fully actuated rigid body systems in three-dimensional space," *Submitted to IEEE Transactions on Control Systems Technology*, 2015.

[28] R. Skjetne, T. I. Fossen, and P. V. Kokotović, "Robust output maneuvering for a class of nonlinear systems," *Automatica*, vol. 40, no. 3, pp. 373–383, 2004.

[29] S. Flixeder, T. Glück, M. Böck, and A. Kugi, "Combined path following and compliance control with application to a biaxial gantry robot," in *Proceedings of the IEEE Conference on Control Applications*, Antibes, France, 2014, pp. 796–801.

[30] A. Isidori, *Nonlinear Control Systems*, 3rd. New York, USA: Springer, 1995.

[31] C. Nielsen and D. Wang, "Path following using transverse feedback linearization: application to a maglev positioning system," in *Proceedings of the American Control Conference*, St. Louis, USA, 2009, pp. 3045–3050.

[32] E. Kreyszig, *Differential Geometry*, 1st, ser. Dover Books on Mathematics. New York, USA: Dover Publications, 1959.

[33] R. Clavel, "Delta, a fast robot with parallel geometry," in *Proceedings of the 18th International Symposium on Industrial Robots*, Lausanne, Switzerland, 1988, pp. 91–100.

[34] P. Guglielmetti, "Model-based control of fast parallel robots," PhD thesis, Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne, Switzerland, 1994.

[35] K. Graichen and B. Käpernick, *A Real-Time Gradient Method for Nonlinear Model Predictive Control, Frontiers of Model Predictive Control*. Shanghai, China: InTech, 2012.

[36] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia, USA: Society for Industrial and Applied Mathematics, 2000.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß des Code of Conduct - Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, August 2016

Michael Schwegel