Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Atussa Koushan

Hybrid Obstacle Aided Locomotion in Snake Robots

Master's thesis in Cybernetics and Robotics

Supervisor: Øyvind Stavdahl

June 2020

NTNU
Norwegian University of Science and Technology

Atussa Koushan

Hybrid Obstacle Aided Locomotion in Snake Robots

Master's thesis in Cybernetics and Robotics
Supervisor: Øyvind Stavdahl
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

Master's Thesis

Studentens navn: Atussa Koushan

Fag: Engineering Cybernetics

Tittel (norsk): Hybrid hindringsbasert fremdrift (HOAL) av slangeroboter

Tittel (English): Hybrid Obstacle Aided Locomotion (HOAL) in Snake Robots

Description:

A number of different principles for snake robot locomotion have been proposed and tested, many of which are based on heuristic rules and stiff position controlled joints. A more physics-based and compliant method is being developed which is based on the formalism of *hybrid position/force control* (HPFC).
In this assignment you will study the differences between manipulator control and the free-ranging snake robot locomotion problem in the context of so-called *Dynamic HPFC* (DHPFC), adapt the method to the new problem as necessary, and assess the results in idealized scenarios.

1. Give a brief overview of the most important strategies for terrestrial snake robot locomotion, and discuss the prospective properties of HPFC in the context of your findings.
2. Adapt the method of *Dynamic Hybrid Position/Force Control*, introduced by Yoshikawa et al. in 1987, to the snake robot locomotion problem, and give a thorough description of the differences and similarities of the snake robot case vs. the traditional manipulator case. Pay special attention to the roles played by the so-called *force* and *position spaces*, and how these can be utilized and further decomposed to illuminate necessary and sufficient conditions for propulsion.
3. Select a set of scenarios and a suitable control structure and simulate these on a suitable platform. The scenarios should be chosen to illustrate and evaluate the central aspects of the method.

Veileder(e): Øyvind Stavdahl, Institutt for teknisk kybernetikk

Trondheim, 13.01.2020

Øyvind Stavdahl
Faglærer

Abstract

Snake robots are serial link robots able to traverse a variety of different terrains. **Obstacle Aided Locomotion (OAL)** is a method for traversing terrains prone by obstacles like rocks and walls which a snake robot can utilize to push itself forward while following a predetermined path. In order to control the resulting obstacle contact forces and snake robot shape simultaneously, the **Hybrid Position/Force Control (HPFC)** method is studied. The combination of OAL and HPFC is referred to as **Hybrid Obstacle Aided Locomotion (HOAL)**.

This report is focused on the dynamic HPFC method, which integrates the robot dynamics into the control. The method is thoroughly studied and adapted to the snake robot case. Furthermore, some additions to the method are proposed and tested. The simulator SnakeSIM, which is based in the Robotic Operating System (ROS), has been used for all experiments. An explanation and evaluation of the simulation framework is provided in the report.

It has been found that the simultaneous control of force and position for a snake robot with several contact points is achievable, but highly dependent on the number of snake robot links and contact points. More specifically, the

experiments suggest that at least two actuated joints should be located between each contact point for successful control. Because the testing was limited to a snake robot with only six links and very simple control goals, it is believed that this number might be higher for more complex scenarios.

Sammendrag

Slangeroboter er seriekoblede roboter som evner å traversere en mengde forskjellige terreng. **Obstacle Aided Locomotion (OAL)** eller hindringsbasert fremdrift er en metode for traversering av terreng preget av mye steiner eller liknende hindringer som slangeroboten kan utnytte for å dytte seg selv fram langs en forhåndsdefinert bane. For å kunne styre de resulterende kontaktkreftene fra hindringer og slangerobotens form samtidig er **Hybrid Position/Force Control (HPFC)** eller hybrid posisjons- og kraftstyring studert. Kombinasjonen av OAL og HPFC er omtalt som **Hybrid Obstacle Aided Locomotion (HOAL)**.

Denne rapporten fokuserer i hovedsak på den dynamiske HPFC-metoden der robotdynamikken er integrert i kontrollen. Metoden er studert grundig og nøye tilpasset slangerobottilfellet. Videre er noen forbedringer til kontrollmetoden foreslått og testet. Simulatoren SnakeSIM, som bygger på rammeverket Robotic Operating System (ROS), er tatt i bruk for utførelsen av alle eksperimenter. En forklaring og evaluering av simulatoren er gitt i rapporten.

Det ble bemerket at hybrid styring av kraft og posisjon for en slangerobot med flere kontaktpunkter er oppnåelig, men høyst avhengig av antallet slangerobotledd og kontaktpunkter. Eksperimentene antyder at minst to aktuerte ledd

mellom hvert kontaktpunkt er nødvendig for tilfredsstillende kontroll. Fordi testingen var begrenset til en slangerobot med kun seks lenker og veldig enkle kontrollmål, så er det antatt at dette minstekravet kan være høyere for mer komplekse scenarioer.

Preface

This report covers the master's thesis for the Cybernetics and Robotics study program at the Norwegian University of Science and Technology (NTNU). The work is a continuation of the independent project work conducted during the previous term, and some parts of this report are taken directly from that work. This is stated explicitly in the concerning sections. Furthermore, a lot of the ideas behind this work are inspired by my supervisor Stavdahl's earlier notes and the discussions we have had on the topic.

The simulator used in this project has been developed by Sanfilippo at NTNU with considerable contribution from skilled students. The code for the method developed in this project is written and integrated to the simulator by me. The figures presented throughout the report are also designed by me, unless stated otherwise.

The unfortunate COVID-19 situation that arose early this year has certainly had an impact on the project flow. There has been a significant lack of theoretical discussions and brainstorming with fellow students, which for me is something that usually has a very positive effect on my work. Luckily, I have still had digital meetings with my supervisor throughout the entire semester

and I want to praise him for his continuous engagement and availability.

Atussa Koushan
28.05.2020
Trondheim, Norway

Contents

Nomenclature

The following list describes several symbols and abbreviations used in the report. All units follow the SI unit system.

Abbreviations

HPFC	Hybrid Position/Force Control
OAL	Obstacle Aided Locomotion
HOAL	Hybrid Obstacle Aided Locomotion
CKC	Closed Kinematic Chain

Control symbols

K_p	Proportional gain
K_i	Integral gain
$\boldsymbol{\tau}_c$	Joint motor control torques
$\boldsymbol{\tau}_P$	Joint torques for desired position control
$\boldsymbol{\tau}_F$	Joint torques for desired force control

Robot dynamics symbols

$\boldsymbol{\tau}$	Generalized torques

$\mathbf{M}$	Mass matrix
$\mathbf{C}$	Coriolis matrix
$\mathbf{g}$	Gravity matrix
L	Lagrangian
K, P	Kinetic and potential energy
I	Moment of inertia of rod
$\mathbf{E}_F$	Task frame axes of force constraints
$\mathbf{E}_P$	Task frame axes of position constraints

Robot kinematics symbols

n	Number of links
n_c	Number of obstacles in contact
N	Number of generalized coordinates
m	Link mass
l	Link length
$\mathbf{q}$	Generalized coordinates
ϕ_i	Joint angle of link i relative to preceding link
θ_i	Angle of link i relative to the base frame
(x_0, y_0)	Position of tail in base frame
(x_i, y_i)	Position of endpoint of link i in base frame b
$(x_{c,i}, y_{c,i})$	Position of contact point on link i in base frame b
k	Distance from joint to contact point on proceeding link

$\mathbf{T_{bi}}$	Transformation matrix from base frame to frame of link i
$\mathbf{T_{bci}}$	Transformation matrix from base frame to frame of contact point on link i
$\mathbf{v_i}$	Velocity of endpoint of link i in base frame b
$\mathbf{v_{c,i}}$	Velocity of contact point on link i in base frame b
$\mathbf{J}_t$	Task Jacobian
$\mathbf{J}_c$	Constraint Jacobian
$\mathbf{r}_t$	Task space coordinates
$\mathbf{r}_c$	Constraint space coordinates

Spaces

$\mathbb{F}$	Force space
$\mathbb{M}$	Motion space
$\mathbb{C}$	Constraint space
$\mathbb{P}$	Propulsion space
$\mathbb{S}$	Shape space
$\mathbb{R}$	Space of real numbers

List of Tables

List of Figures

Chapter 1

Introduction

This chapter presents the most relevant previous research contributions to the studied snake robot locomotion method and the hybrid position/force control method. Furthermore, the scope of the project and its contributions are summarized. The simplifications and limitations, as well as the snake robot model description can also be found in this chapter. Lastly, the structure of the report is described.

1.1 Previous work

The department of engineering cybernetics at the Norwegian University of Science and Technology (NTNU) has made significant contributions to the field of snake robot control, related to both aquatic snake-like propulsion and efficient snake robot locomotion on flat surfaces [1].

Some common modes of snake robot locomotion on flat surfaces are lateral

undulation, concertina locomotion and sidewinding. Lateral undulation and sidewinding are dependent on ground friction for the snake robot to propel itself forward, whereas concertina locomotion utilizes walls or narrow spaces that the snake robot can anchor itself against and push itself forward [2]. Terrestrial snake robot locomotion can also be achieved in terrains with more defined irregularities, such as rocks. In 2011, Liljebäck et al. [3] proposed a locomotion strategy for continuously adapting the shape of the snake robot to the environment to avoid obstacle jams during straight line path following. Directional compliance, proposed by Wang et al. [4] in 2020, is a novel method that adapts the snake robot stiffness so that it conforms to the environment in some directions and resists in others and thus uses obstacles to enhance the existing propulsion mode. Both of these strategies are reactive methods that consider obstacles already encountered.

Obstacle Aided Locomotion (OAL) is on the other hand a method that uses obstacles as the main part of the locomotion strategy and always seeks to find the obstacles that can profit the propulsion the most. The idea of this method, which was introduced by Transeth et al. [5] in 2008, is that the snake robot pushes against obstacles in order to propel itself forward along a predefined path. The principle is comparable to lateral undulation, where ground friction is used instead of obstacles to push against. It is thus possible to say that OAL is a very discrete special case of lateral undulation. For this reason it is believed that solutions and findings related to OAL can later be used to profit and optimize snake robot locomotion gaits like lateral undulation.

In OAL it is desired to control the force against obstacles and shape of the robot independently. Thus, Stavdahl [1] proposed a combination of OAL and *Hybrid Position/Force Control (HPFC)*, leading to the term *Hybrid Obstacle Aided*

Locomotion (HOAL). Klafstad [6] summarized the theory around this concept in 2018, and it was further investigated and tested by the author in her previous project work [7] in 2019, both with emphasis on the HPFC method introduced by West and Asada [8] in 1985. This method was developed after the concept was first introduced by Raibert et al. in 1981 [9], and aims at controlling constrained robot manipulators based on kinematic projections of the motion and force such that these variables comply with both the allowable and desired behaviour of the snake robot. Yoshikawa [10] advanced the method in 1987 by including the dynamics of the robot in the control calculations.

The research of HPFC has mainly been focused on the control of traditional robot manipulators. Nansai et al. [11] applied HPFC on snake robots with wheels in 2016. The wheels restrict direct sideways movement of the robot and can therefore be considered as obstacles on the sides of the snake robot. The HPFC was however not applied for the locomotion purpose of the robot, but for a typical robot manipulator task the snake robot was set to execute. A lot of work still remains when it comes to combining snake robot OAL with HPFC. Because the behaviour of the snake robot is driven by its dynamics, the research is now continued to explore the dynamic HPFC method in light of HOAL.

1.2 Scope of the project

1.2.1 Thesis assignment interpretation

The detailed assignment for this thesis is provided in the very beginning of the report. The author's perception is that the main focus lies on mathematically

adapting the dynamic HPFC method of Yoshikawa [10] to the snake robot case with special attention to the snake robot force and position spaces. A control structure should then be designed and implemented in a carefully selected simulator to conduct experiments that help evaluate important aspects of the developed method. A discussion of how the dynamic HPFC method can be used to achieve HOAL should also be carried out.

1.2.2 Contributions

The most significant contributions from this project are listed below.

- A mathematical adaptation of the dynamic HPFC method for snake robots intended to perform HOAL.
- An analysis of further control requirements, as well as suggestions to associated improvements and modifications to the dynamic HPFC method.
- A discussion of the requirements to the snake robot itself and its configuration relative to the obstacles it is in contact with for successful dynamic HPFC and HOAL.
- Thorough selection of a suitable simulator, as well as a review of this simulator and a comparison to other corresponding simulators.
- A guiding description of the simulator structure and how it generally can be utilized.
- Implementation and testing of the the developed snake robot model and dynamic HPFC method in the simulation environment.
- A discussion of the findings in light of HOAL.

- Proposals for further improvements to the snake robot dynamic HPFC method and control structure.
- Suggestions to some key aspects for achieving HOAL and a discussion of these.

1.3 Model specifications

This section presents the physical aspects of the snake robot model used throughout the report. The model has three main parts, namely the environment, the snake robot and the obstacles. The value of all variables needed to express the model and control goals are assumed known at all times. Further simplifications and assumptions are presented below. The parts *Assumptions* and *Further model description* are taken from the previous work of the author [7], but some of the assumptions are modified to conform with the current project.

1.3.1 Simplifications

Snake robots performing OAL are typically in contact with several obstacles at a time and it is thus desired to control the snake robot at several contact points simultaneously. A higher number of joints and therefore actuators are beneficial for execution of this kind of control. However, in order to reduce the dimensions of the dynamic and kinematic calculations, the snake robot model used for testing consists of six links and five joints, which has limited the possible test variations.

The computed snake robot model is designed for a very specific scenario in which it is assumed that there are three contacts between a predefined set of

snake robot links and obstacles at all times. Consequently, all movement of the snake robot can only be performed in the bounds of this assumption, meaning at most one link length displacement.

Another simplification made in the mathematical model is that the snake robot and obstacles are assumed to be 2D bodies, although they have 3D properties in the simulator. However, since all movement and control happens in the 2D plane it is not considered to have been a very evident deficiency.

The collisions between the snake robot and obstacles are not modeled because it is assumed that the contact is maintained at all times. However, small collisions are experienced by the simulator as a result of a sensitive force sensor and the contact point on a link moving within the range of the link.

1.3.2 Assumptions

Assumptions 1-6 are taken from Stavdahl [1], whilst assumptions 7-9 are specific for this project.

1. All parts of the model are assumed to be rigid.
2. The robot has n joints and all links have length l and mass m.
3. Only flat, 2-dimensional cases are considered.
4. The robot has no lateral extension.
5. There is no friction.
6. Obstacles have no spatial extent, only a static position in the plane.
7. Any link is in contact with at most one obstacle at the time.

8. The robot is in touch with obstacles in its start configuration.

9. Contact between a link and an obstacle is maintained.

1.3.3 Further model description

The snake robot

The snake robot itself is modeled as a simple planar robot manipulator with links and joints. The main difference between the snake robot and a classic robot manipulator is the property that the snake robot is not physically attached to any fixed point in the world. This frees one constraint. However, it is still relevant to express the position of the first link of the snake, also denoted as the tail. This is performed by introducing three virtual joints to the model; two translational and one rotational joint. These joints are not controllable and merely for describing the kinematics, dynamics and constraints. The model of the snake robot is visualized in Figure 1.1, where physical robot links are blue and the virtual ones are green.

Figure 1.1: Model of snake robot with n links

The environment

The environment is the (x,y)-plane in Figure 1.1, and consists of nothing but the robot and the obstacles.

The obstacles

The obstacles are modeled as rigid points in the plane and any contact with them is considered a point contact. Figure 1.2 shows three obstacles in contact with a snake robot.

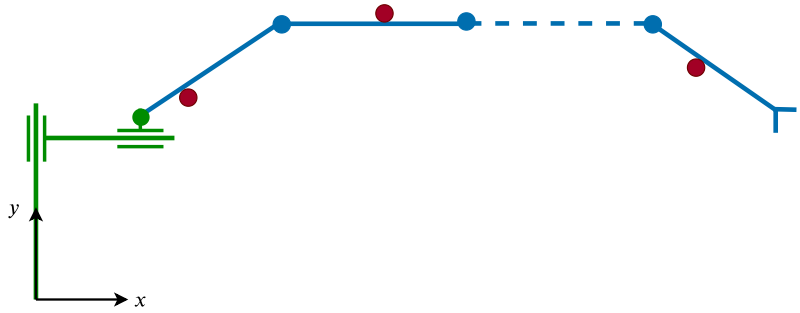


Figure 1.2: Model of snake robot and obstacles

1.4 Report structure

It is assumed that the reader is familiar with basic robotics and control theory. A more thorough explanation of the topics can be found in [12], [13], [14], [2].

To begin with, the report introduces a set of different terrestrial snake robot locomotion strategies in Chapter 2. This chapter also covers the required mathematical background theory for understanding the dynamical HPFC

method. The dynamic HPFC method is then thoroughly explained in Chapter 3. All related considerations and limitations are also provided here and the method is put in the context of HOAL. Chapter 4 explains the structure of the chosen simulator and how it is used for the simulated experiments, which are presented in Chapter 5. The results from the experiments and general analysis of the dynamic HPFC method on snake robots are discussed in Chapter 6. Lastly, Chapter 7 concludes the work and Chapter 8 proposes some ideas for future research and possible improvements based on the challenges encountered in this project.

Chapter 2

Background theory

This chapter gives an overview of some of the most important strategies for terrestrial snake robot locomotion to address the first part of the thesis project description. Section 3.3 in Chapter 3 further discusses the use of HPFC together with these locomotion strategies. The rest of this chapter focuses on theory required to adapt the method of dynamic HPFC to the snake robot model. This includes an explanation of the snake robot kinematics, dynamics and constraints.

Some parts of this chapter are fully or partly taken from the previous project work of the author [7]. This is stated in the respective sections.

2.1 Terrestrial snake robot locomotion strategies

Several methods have been proposed for snake robot locomotion, most of which are inspired by nature. This part presents some of the well established

strategies, as well as the newer approach, obstacle aided locomotion, which this report is based on. Finally, compliance control in snake robot locomotion is presented to open up for a comparison to what the hybrid position/force control offers.

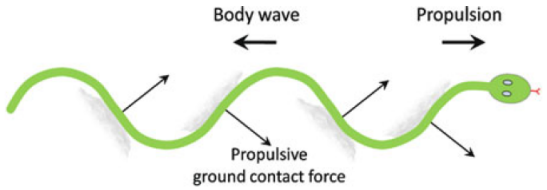
2.1.1 Traditional locomotion strategies

Lateral undulation is the fastest and by far the most commonly implemented locomotion gait for snake robots [15]. It is a continuous movement of the entire body of the snake relative to the ground. The locomotion is obtained by propagating sine-waves from the front to the rear of the snake [16], as illustrated in figure 2.1a. Successful propulsion by lateral undulation requires that the snake has some grip to the surface enabling it to glide forward without slipping sideways [17]. This again puts requirements on the friction coefficients, namely that the coefficient in lateral direction have to be greater than in the forward direction. If the movement is conducted without slipping, every part of the body will pass through the same points on the ground.

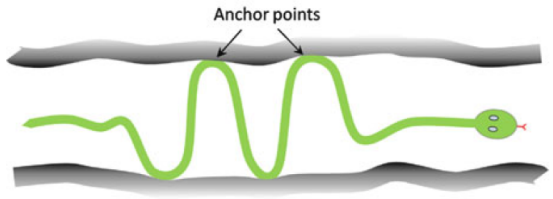
Concertina locomotion is not employed on open ground, but rather in narrow spaces where the available range of motion is limited. The motion is carried out by curving the body to create an anchor against the narrow environment (see figure 2.1b). The snake alternates between curving the back and front part of the body so that the front part can be stretched forward and the back part can be drawn up [2].

A resembling mode of locomotion is **sidewinding**, in which one part of the body acts as an anchor while the other part is moved forward. The part that moves forward is simply lifted off the ground and displaced while the other

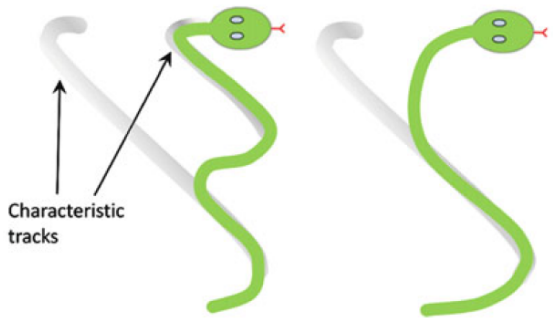
part stays put, typically in areas with loose sand [2]. It can be seen as a kind of spiraling motion. Figure 2.1c gives a visual explanation of this gait.



(a) Lateral undulation



(b) Concertina locomotion



(c) Sidewinding

Figure 2.1: Illustration of some traditional locomotion strategies [2].

2.1.2 Central pattern generators (CPGs)

The information presented in this section is taken from [18].

A central pattern generator for locomotion control is where neuroscience meets robotics. CPGs consist of neurons that produce an oscillating, rhythmic output without requiring sensory information. In nature, CPGs play an important role in periodic actions like breathing, walking and other modes of locomotion. While sensory feedback is not needed for generating the rhythms, it plays a very important role in shaping the rhythmic patterns. This is fundamental for keeping CPGs and body movements coordinated.

The method produces stable rhythmic patterns and is able to rapidly return the system to its normal rhythmic behavior after transient perturbations. Furthermore, CPG models tend to reduce the dimensionality/complexity of the control problem in that they have only a few control parameters. These parameters allow modulation of the locomotion, for instance the speed and direction or even the type of gait. This allows for higher-level controllers to circumvent the generation of multidimensional motor commands and stick to higher level control signals.

CPGs can be applied to cyclic locomotion gaits like lateral undulation, sidewinding and concertina locomotion since they are based on rhythmic patterns.

2.1.3 Obstacle-aided locomotion (OAL)

Aforementioned locomotion strategies are first and foremost successful in plain, obstacle-free environments where ground friction can be used to achieve propulsion. The OAL method, on the other hand, considers flat environments

with obstacles. The obstacles can be compared to the ground friction in other locomotion gaits like lateral undulation. This is because both obstacles and ground friction are used to push against. Thus, OAL can be seen as a kind of discrete special case of lateral undulation. The rest of the explanation of OAL is taken from the project work of the author [7].

Instead of avoiding physical contact between the robot and obstacles, obstacle aided locomotion aims at profiting from it by using the obstacles as push-points to propel itself forward. This concept is illustrated in Figure 2.2. OAL was first introduced by Transeth et al. in 2008 [5]. The motivation behind this method was based in the ability of biological snakes to utilize irregularities in the terrain for more efficient locomotion.

Liljebäck et al. [2] describe two major challenges related to OAL:

1. It is unknown in advance when and where the snake robot will make contact with its environment.

2. The development of a strategy for adjusting the shape of the robot so that forward propulsion is achieved in any given contact situation.

The following hypothesis is also stated in [2].

> *Obstacle-aided snake robot locomotion can be achieved by producing body shape changes where the links in contact with obstacles are rotated so that the components of the contact forces in the desired direction of motion are increased.*

Holden et al. [19] address the second challenge by formulating an optimization problem that seeks to minimize energy consumption while achieving propulsion along a user-defined desired path. The output of this optimization

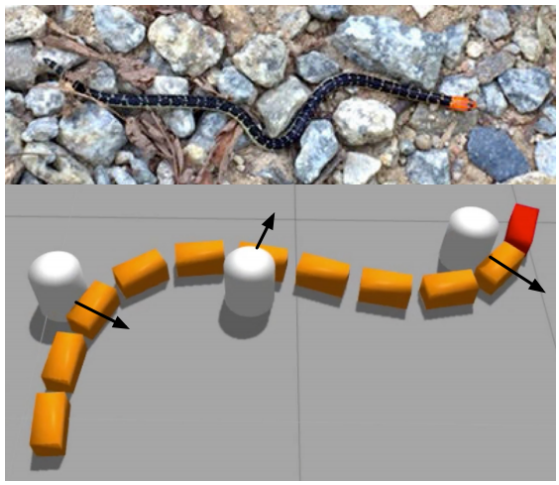


Figure 2.2: Obstacle-aided locomotion illustration [21]

is the optimal motor torque inputs. In addition to a user-defined path, this method assumes that the desired link angles at the obstacles are given.

Bayraktaroglu et al. [20] mention that only the trajectory of the leading link should be arbitrarily determined. Moreover, Bayraktaroglu et al. [20] state that in a steady smooth motion, the trajectory of the leading link must be computed as a function of available push-points for the next contact, and the desired position and orientation of the following links are those that mimic the motion of the leading link.

2.1.4 Locomotion strategies with compliance control

Biological snakes cannot in any way be seen as rigid animals. Robots, on the other hand, are by default very rigid, both in the body material and actuators used. Compliant control is fundamental when dealing with unstructured envi-

ronments because it implicitly controls the energy transfer to the environment [22].

Two main methods can be applied to snake robots to mimic the compliant and adaptive behavior between snakes and the environment they traverse. The first method is using a compliant controller for the actuators, in which they adapt a spring-like behavior. The shape-based compliance and directional compliance, among others, go under this category and are explained here. The second method is directly designing the robot with compliance, meaning that the mechanical parts are compliant.

Shape-based compliance

A very common way of controlling the shape of snake robots has been specifying the motion of each individual joint directly. This works well as long as the snake robot is traversing a flat surface without obstacles that would lead to great deviations from the planned motion. In environments with obstacles, on the other hand, Liljebäck et al. [23] proposed a specification of the motion of the robot in terms of *shape control points (SCPs)*. These points specify which coordinates it is desired that the snake goes through. The points are further connected with a shape curve defined by an interpolation technique. The desired joint angles of the snake robot are eventually found by aligning a virtual robot with the shape curve and adapt its joint angles.

The work of [23] is based on a model of the snake robot crawling by lateral undulation on a flat surface with circular obstacles. The compliant behavior of the snake robot is then achieved by assigning mass-spring-damper dynamics to the SCPs, making the SCPs and thus the shape curve compliant. [23] propose that the adaptive behavior of the shape curve should depend on the direction of

the contact force with respect to the forward direction of motion. Consequently, a natural choice is to make the shape curve highly compliant for obstructive contact forces and less compliant for propulsive contact forces.

Directional compliance

The disadvantage of shape-based compliance is that not all terrain features are helpful for locomotion. Some might impede the movement of the robot instead of aiding it. Because of this, a novel approach called directional compliance has been proposed by Wang et al. [4]. The approach is an improved version of the shape-based compliance controller that adapts the compliance according to the information it has about the obstacles. This information is obtained from the force measured through the joints of the robot. Thus, it is what can be called a reactive controller. Directional compliance allows the robot to selectively admit forces applied on one side and reject forces from the other side [4].

The most substantial discovery of Wang et al. [4] is that the amplitude of the shape parameters act as an indication of the robot's forward progression. Thus, the amplitude difference from the nominal value is used to determine whether the robot should increase or decrease it's curvature at certain segments of its body. Wang et al. [4] denote admitting external forces that result in increases in curvature as positive directional compliance and admitting forces that result in decreases in curvature as negative directional compliance. The known shape-based compliance is the nominal compliance of the controller. Furthermore, a filter function is used to adapt this nominal compliance and include the two other modes.

In summary, directional compliance enables the robot to either push away from, or comply to, terrain features, based on the torque feedback measured

over time [4]. This dynamical system strategy was proven successful even in a three dimensional environment. However, the main aim of this controller is utilizing obstacles the snake robot "by chance" collides with in order to avoid getting stuck between obstacles, rather than following a specific path. This means that the method is unlikely to choose the most optimal path to move forward, as is desired in OAL.

Mechanical compliance

When a robot is mechanically compliant, it typically means that it has flexible or soft mechanics. Several technologies have been explored to realise this property, like pneumatic, hydraulic, polymers etc. [22]. According to Calanca [22], most of the implementations at the current state of the art make use of traditional electric motors with the addition of a soft element and/or a compliant controller.

Mechanical compliance allows for implicitly controlling the power transferred to the environment and to exhibit displacement if a force is applied. Furthermore, it is obvious that the response time of a mechanical compliant system always will be faster than a system with a compliant controller. The relation of compliance control to position and force control, is that an ideal position controlled system has zero compliance whilst an ideal force controlled system has infinite compliance [22].

2.2 Snake robot kinematics

This section is a modified part of the previous work of the author [7]. The modifications are:

- re-made figures
- variable notations
- fixed mistake in (2.5)
- altered vector of generalized coordinates for the constained case in 2.2.1.

The snake robot is modeled as a serial chain, which is a system of rigid bodies in which each member is connected to two others, except for the first and last members that are each connected to only one other member [14]. As opposed to traditional robot manipulator models, the first joint in the snake robot model is not physically connected to a base.

The vector of generalized coordinates $\mathbf{q}$ for a snake robot with n links is

$$\mathbf{q} = [\phi_0 \quad \phi_1 \quad ... \quad \phi_{n-1} \quad x_0 \quad y_0]^T. \tag{2.1}$$

The coordinates (x_0, y_0) and ϕ_0 represent the position and orientation of the tail of the snake robot in reference to the base frame (x, y). These coordinates cannot be directly controlled and will therefore be referred to as virtual coordinates. The generalized coordinates $\phi_1, ..., \phi_{n-1}$, corresponding to the actuated joints, refer to the angle of the following link relative to the preceding link. The number of generalized coordinates including two position coordinates and n joint angles is $N = n + 2$.

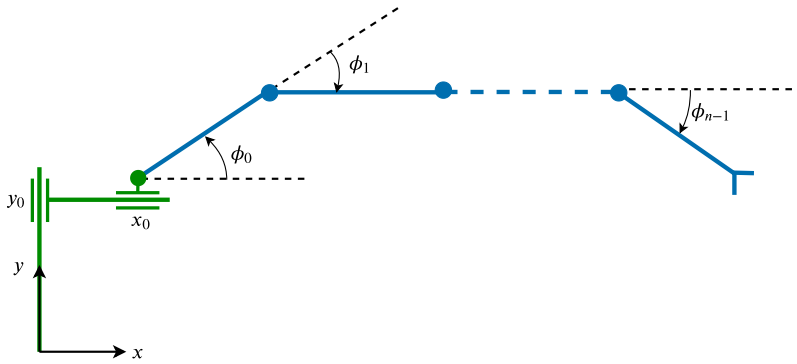


Figure 2.3: Model of snake robot with notation

The model of the snake robot with the named variables are illustrated in Figure 2.3.

Homogeneous transformation matrices are used to express the pose (position and orientation) of the links in relation to the base frame. This means that as long as the joint angles and size of the snake robot are known, the Cartesian positions can be calculated. The homogeneous transformation matrix for the end point of link i from the base frame b is given by (2.2). The base frame will stay put regardless of motion of the robot. Note also that the link length l is assumed equal for all the links.

$$\mathbf{T}_{bi} = \mathbf{D}_x(x_0)\mathbf{D}_y(y_0)\sum_{k=0}^{i-1}\mathbf{R}_z(\phi_k)\mathbf{D}_x(l) \tag{2.2}$$

The translation and rotation matrices are given by

$$\mathbf{D}_x(x) = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{D}_y(y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.3}$$

$$\mathbf{R}_z(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The transformation matrix from the reference frame to the center of link i can be found in the same manner. The only difference is that the very last translational matrix has to take the argument $l/2$ instead of l. This is useful to keep in mind as it will be used in Section 2.3 for the derivation of the kinetic energy of the links.

As mentioned earlier, the transformation matrix $\mathbf{T}_{bi}$ can be used to find the absolute orientation and position of the tip of link i in the base frame. The resulting matrix can be written on the form

$$\mathbf{T}_{bi} = \begin{bmatrix} \mathbf{R}_{bi}(\theta_i) & \mathbf{t}^r_{ri} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{2.4}$$

The position is directly extracted from $\mathbf{t}^r_{ri} = [x_i y_i]^T$. The orientation θ_i is found by comparing $\mathbf{R}_{bi}$ to $\mathbf{R}_z$ and solving for θ_i.

The global link orientations θ_i are visualized in Figure 2.4.

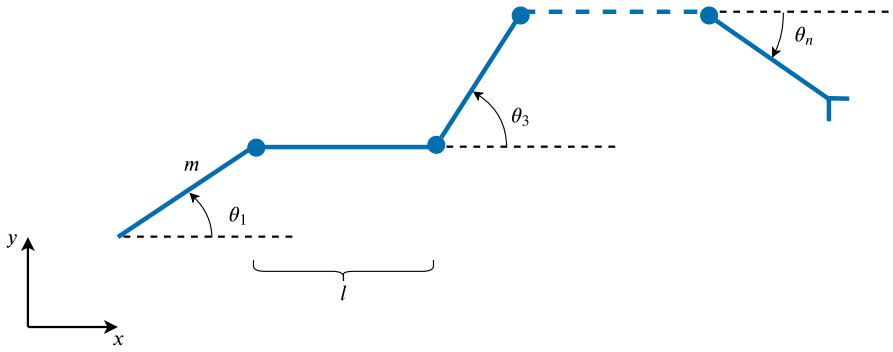


Figure 2.4: Snake robot global link angles

Alternatively, one can directly compute the position of the center of a link i from the expressions below

$$\begin{aligned} x_i &= x_0 + \sum_{k=0}^{i-2} l \cos\left(\sum_{j=0}^{k} \phi_j\right) + \frac{l}{2} \cos\left(\sum_{j=0}^{i-1} \phi_j\right) \\ y_i &= y_0 + \sum_{k=0}^{i-2} l \sin\left(\sum_{j=0}^{k} \phi_j\right) + \frac{l}{2} \sin\left(\sum_{j=0}^{i-1} \phi_j\right), \end{aligned} \tag{2.5}$$

where $1 \leq i \leq n$.

Forward and inverse instantaneous kinematics

The well known Jacobian lets us transform between Cartesian and joint velocities. It is derived by taking the partial derivative of the x and y position of link $1 \leq i \leq n$ with respect to all generalized coordinates

$$\mathbf{J_i} = \begin{bmatrix} \frac{\partial x_i}{\partial q_1} & \cdots & \frac{\partial x_i}{\partial q_{N-1}} & \frac{\partial x_i}{\partial q_N} \\ \frac{\partial y_i}{\partial q_1} & \cdots & \frac{\partial y_i}{\partial q_{N-1}} & \frac{\partial y_i}{\partial q_N} \end{bmatrix}. \tag{2.6}$$

The relationship between the Cartesian velocity $\mathbf{v}$ of the point (x_i, y_i) on the robot and the joint velocities $\dot{\mathbf{q}}$ can thus be written as

$$\mathbf{v_i} = \mathbf{J_i}(\mathbf{q})\dot{\mathbf{q}} \quad \text{and} \quad \dot{\mathbf{q}} = \mathbf{J_i}(\mathbf{q})^{\dagger}\mathbf{v_i}. \tag{2.7}$$

The first equation is formally referred to as the forward instantaneous kinematics, whereas the second one is referred to as the inverse instantaneous kinematics. $\mathbf{J}(\mathbf{q})^{\dagger}$ is the Moore-Penrose [24], [25] pseudo inverse of the Jacobian, which has to be used as a result of the Jacobian being non-square.

2.2.1 Constrained kinematics

For the case in which the robot is in contact with the environment, the motion will be constrained. The obstacles found in the environment are modelled as single frictionless points. The only constraint imposed by the environment is that the robot cannot penetrate the obstacles. It can, however, both apply an arbitrary large force against them or move along them.

The model assumes that any link can be in contact with at most one obstacle at the time. To represent the mentioned constraint, the vector of generalized coordinates is expanded with $2n_c$ further elements, where n_c is the number of obstacles in contact. The updated vector $\mathbf{q}$ is now

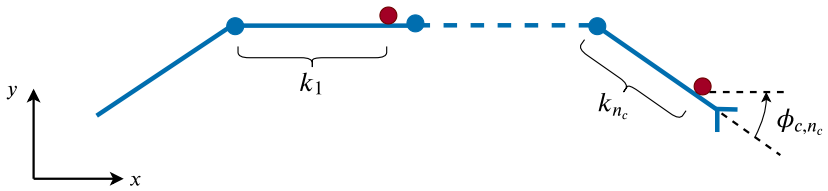


Figure 2.5: Snake robot contact parameters

$$\mathbf{q} = [\phi_0 \quad ... \quad \phi_{n-1} \quad x_0 \quad y_0 \quad k_1 \quad ... \quad k_{n_c} \quad \phi_{c,1} \quad ... \quad \phi_{c,n_c}]^T, \tag{2.8}$$

where $N = n + 2 + 2n_c$ is the new number of generalized coordinates.

The newly introduced coordinates $k_1, ..., k_{n_c} \geq 0$ represent the distance to the contact point from the preceding joint to the link in contact. For instance, if link 2 is the first link in contact with an obstacle, then the coordinate k_1 is the distance between the joint connecting link 1 and 2 and the contact point measured along the second link, as illustrated in Figure 2.5. Furthermore, $\phi_{c,1}, ..., \phi_{c,n_c}$ are the angles from the links in contact to the global x-axis. This variable is later used to describe the constraint that the obstacles are fixed in space. An example is illustrated in Figure 2.5 for the last link in contact with an obstacle. Seeing as there is no actuation force directly connected to the obstacle-related coordinates, they will be referred to as virtual joints or virtual coordinates.

The position of a contact point $1 \leq j \leq n_c$ on link $1 \leq i \leq n$ in the base frame can be derived through the corresponding transformation matrix (2.9).

$$\mathbf{T}_{bci} = \mathbf{D}_x(x_0)\mathbf{D}_y(y_0) \sum_{k=0}^{i-2} (\mathbf{R}_z(\phi_k)\mathbf{D}_x(l))\mathbf{R}_z(\phi_{i-1})\mathbf{D}_x(k_j) \tag{2.9}$$

Constrained instantaneous kinematics

The Jacobian matrix related to the velocity of the contact point can be derived in the same manner as in the unconstrained case. The only difference is that the partial differentiation of the contact point (x_c, y_c) is now taken with respect to the extended vector of generalized coordinates (2.8). The resulting contact Jacobian for a contact point on link $1 \leq i \leq n$ is thus

$$J_{c,i} = \begin{bmatrix} \frac{\partial x_{c,i}}{\partial q_1} & \cdots & \frac{\partial x_{c,i}}{\partial q_{N-1}} & \frac{\partial x_{c,i}}{\partial q_N} \\ \frac{\partial y_{c,i}}{\partial q_1} & \cdots & \frac{\partial y_{c,i}}{\partial q_{N-1}} & \frac{\partial y_{c,i}}{\partial q_N} \end{bmatrix}. \tag{2.10}$$

This Jacobian will end up being quite sparse, seeing as the coordinate of a contact point is independent of all other contact coordinates. This is a property that can be exploited using sparse solvers if the snake robot has a large number of links.

The relationships between the Cartesian velocity of a contact point on link $1 \leq i \leq n$ and the joint velocities can now be expressed as

$$\mathbf{v_{c,i}} = \mathbf{J_{c,i}}(\mathbf{q})\dot{\mathbf{q}} \quad \text{and} \quad \dot{\mathbf{q}} = \mathbf{J_{c,i}}(\mathbf{q})^{\dagger}\mathbf{v_{c,i}}. \tag{2.11}$$

2.3 Snake robot dynamics

This section, similar to the last section, is taken from [7]. The only modifications made are the variable notations and an error fix in (2.19).

The snake robot has $n-1$ joint actuators that all can apply torques to their corresponding joints. The dynamics describe how the robot moves in response

to these actuator forces. For simplicity, it is assumed that the actuators do not have dynamics of their own and, hence, arbitrary torques can be commanded at the joints of the robot [26].

The dynamics of the snake robot will be expressed using the joint space equations of motion formulation

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}. \tag{2.12}$$

Because the movement is restricted to the 2D plane, the gravitational term $\mathbf{g}(\mathbf{q})$ can be neglected and the equations of motion can be rewritten to

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau}, \tag{2.13}$$

where $\mathbf{M}(\mathbf{q})$ and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the mass matrix and Coriolis matrix respectively. $\boldsymbol{\tau}$ is the vector of generalized torques corresponding to the generalized coordinates (2.1). Furthermore, the elements corresponding to the virtual coordinates will be zero at all times.

Solving (2.13) with respect to $\ddot{\mathbf{q}}$ yields

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q})(\boldsymbol{\tau} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})). \tag{2.14}$$

Several methods exist for finding the equations of motion for a robot. The Euler-Lagrange method [12], which is chosen here, is based on the difference in kinetic energy (K) and potential energy (P) of the system, also known as the Lagrangian

$$L = K - P. \tag{2.15}$$

The equations of motion can now be expressed as a second order partial differential equation

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} = \boldsymbol{\tau}. \tag{2.16}$$

Again, simplifications can be made from the restricted movement in the world and thus the potential energy P can be neglected. The Lagrangian is therefore simply equal to the kinetic energy, which is the sum of the kinetic energy for every link [27]. Furthermore, the kinetic energy for one link i is divided into two parts, $K_{translational}$ and $K_{rotational}$. The kinetic energy can now be express as

$$K = \sum_{i=1}^{n} (K_{translational,i} + K_{rotational,i}), \tag{2.17}$$

where the translational and rotational kinetic energy is given in (2.18) and (2.19) respectively.

$$K_{translational,i} = \frac{1}{2} m (\dot{x}_i^2 + \dot{y}_i^2) \tag{2.18}$$

Here m is the link mass, and $(\dot{x}_i, \dot{y}_i)$ make out the velocity of the center of the link found by differentiating (2.5) with respect to time.

$$K_{rotational,i} = \frac{1}{2} I \dot{\theta}_i^2 \tag{2.19}$$

$\dot{\theta}_i$ is the rotational velocity of link i with respect to the base frame. Furthermore, every link has the same moment of inertia, namely $I = (1/12)ml^2$. This is the moment of inertia of a rod, corresponding to the moment of inertia of a cylinder with zero radius [12].

2.4 Snake robot constraint formulation

Figure 2.5 illustrates the used model of the snake robot and its environment. When looking at this model, it is evident that the snake robot will experience constraints if it tries to move its links in the directions of the obstacles. One way of expressing this constraint is limiting the velocity of the links in contact. The position of contact on link i was in 2.2.1 denoted as $(x_{c,i}, y_{c,i})$. Thus, the corresponding velocity $\mathbf{v}_{c,i}$ can be written $(\dot{x}_{c,i}, \dot{y}_{c,i})$.

The velocity of this point is not constrained in every direction, but in the direction normal to the link. This follows from the assumption of the contact being a point contact. The normal velocity can be found with the use of the angle θ_i of link i with respect to the base frame. From basic trigonometric laws, the normal velocity is found to be

$$v_{nc,i} = -sin(\theta_i)\dot{x}_{c,i} + cos(\theta_i)\dot{y}_{c,i}. \tag{2.20}$$

Since there can only be one contact per link and the contact can only take place at one side of the link at the time, the link can still move away from the obstacle with a velocity normal to the link. Holden et al. [19] introduce a variable γ_i that holds information about which side of the link the obstacle is positioned.

The variable can take the values $-1, 0, 1$, where $\gamma_i = 1$ for obstacles to the right of the link and vice versa. $\gamma_i = 0$ for links not in contact with any obstacles.

The allowed normal direction of movement for link i can now be expressed as

$$\gamma_i v_{nc,i} \geq 0. \tag{2.21}$$

It might be desired to stay in touch with the obstacle for propulsion purposes. Because of this, it is reasonable to simplify the constraint equation to not let the contact point on the link move in any normal direction to the link at that point. The constraint can thus be expressed as

$$\gamma_i v_{nc,i} = \gamma_i(-sin(\theta_i)\dot{x}_{c,i} + cos(\theta_i)\dot{y}_{c,i}) = 0 \tag{2.22}$$

Moreover, the side of the contact is insignificant if the velocity is limited in both directions. (2.22) is further simplified to

$$v_{nc,i} = -sin(\theta_i)\dot{x}_{c,i} + cos(\theta_i)\dot{y}_{c,i} = 0. \tag{2.23}$$

In robot manipulator theory it is common to use both the base coordinate frame and a coordinate frame related to the end effector. The end effector coordinate system is irrelevant for the contact point velocities. It is however possible to define the velocities in coordinate systems related to the contact points. A coordinate system related to the contact point on link i is $(x_{t,i}, y_{t,i})$, where the x-direction is the direction of the link in contact. This means that

the rotation of this coordinate system relative to the base coordinate system is θ_i. The subscript t stands for *task* and is introduced for its purpose in the dynamic hybrid position controller explained in 3.1.2.

Since the velocity normal to the link in the task coordinate frame is the velocity along the y-axis, the constraint on link i can be rewritten to

$$\dot{y}_{t,i} = 0, \tag{2.24}$$

which is much simpler than (2.23).

Chapter 3

Dynamic HPFC for snake robots

This chapter explains the dynamical HPFC method of [10] adapted to the snake robot case and what is here referred to as the traditional HPFC method, namely the method of West and Asada [8]. Furthermore, the advantages and challenges that come with adapting the dynamical method to a snake robot model are presented and discussed. Some modifications attempting to make the control more robust are also suggested. Lastly, the method is put into the context of HOAL, and further requirements for HOAL are suggested and reviewed.

3.1 Hybrid position/force controllers

The goal of the snake robot is to push against obstacles in a fashion that yields forward propulsion along a path. Consequently, the robot will have to curve itself along the path whilst applying a force to the obstacles considered advantageous. The behavior of the robot has to comprise with the constraints arising from the contact, which further motivates the use of hybrid position/force control (HPFC).

HPFC is not a control method per se, but rather a method for determining when and in which directions force or motion control should be applied. It is desired to control motion along the unconstrained motion directions and force along the constrained motion directions. Different approaches to this problem exist. One is the use of selection matrices, introduced by Raibert and Craig et al. [9]. The disadvantage of this approach is that the directions in which force and motion should be controlled has to be recalculated for every step. In another approach, introduced by West and Asada [8], two projection matrices are used as filters in joint space to automatically select between position and force controlled vectors. There is however a disadvantage to this method as well, which is that the dynamics of the robot are ignored. A thoroughly studied method called dynamic HPFC, developed by Yoshikawa [10], is therefore presented and adapted to the snake robot.

The theory on traditional HPFC up until the part *Passive joints*, as well as this introduction, are a modified part of the earlier project work of the author [7]. The rest of the section is focused on the dynamic HPFC method adapted to the snake robot case.

3.1.1 Traditional HPFC

Like mentioned above, velocity and force can be controlled in the directions in which they are not constrained. The end effector space of a robot can be divided into two orthogonal domains, a position domain and a force domain. These domains are complementary to the directions of the corresponding constraints at the end effector. It is logical to conclude that if there is contact with the environment, motion cannot be controlled freely. On the other hand, if there is no contact, there is no direction in which the robot can apply a force. Ergo, the force and motion control directions do not overlap and the domains are orthogonal. This means that position and force can be controlled independently and arbitrarily in these domains.

The following relationships are based on the Jacobian introduced in 2.2.1.

$$\mathbf{v} = \mathbf{J}\dot{\mathbf{q}}, \quad \boldsymbol{\tau} = \mathbf{J}^T\mathbf{f} \tag{3.1}$$

An important observation is that constraints due to contact with the environment are constraints due to a closed kinematic chain. In general, this is something that occurs when at least *two* points of the robot are in contact with the environment. For the snake robot this might not always be the case. It is however possible to define a virtual closed kinematic chain where the robot is connected to the base with the virtual joint variables x_0, y_0 and ϕ_0. A separate Jacobian is calculated for each closed kinematic chain, as explained in 2.2.1.

Relationship (3.2) comes from the motion being constrained at a contact point.

$$\dot{\mathbf{v}}_{\mathbf{ci}} = \mathbf{J}_{\mathbf{ci}}\dot{\mathbf{q}} = \mathbf{0} \tag{3.2}$$

The solution to (3.2) can be proven to be

$$\dot{\mathbf{q}} = (\mathbf{I} - \mathbf{J}_{\mathbf{ci}}^{+}\mathbf{J}_{\mathbf{ci}})\mathbf{y}, \tag{3.3}$$

where $\mathbf{y}$ can be an arbitrary vector, as it will yield zero end effector motion. Furthermore, since the matrix $\mathbf{J}_{\mathbf{ci}}$ might be non square, the pseudo inverse $\mathbf{J}_{\mathbf{ci}}^{+}$ is used. For a closed kinematic chain, the work done at the end of the chain must also be zero. Therefore, the sum of the work done by each of the joints must be zero:

$$\boldsymbol{\tau}^{T}\dot{\mathbf{q}} = \boldsymbol{\tau}^{T}(\mathbf{I} - \mathbf{J}_{\mathbf{ci}}^{+}\mathbf{J}_{\mathbf{ci}})\mathbf{y} = \mathbf{0}. \tag{3.4}$$

(3.4) has the general solution

$$\boldsymbol{\tau} = (\mathbf{J}_{\mathbf{ci}}^{+}\mathbf{J}_{\mathbf{ci}})^{\mathbf{T}}\mathbf{z}, \tag{3.5}$$

where $\mathbf{z}$ can be an arbitrary vector.

The allowable motion is now characterized by $[\mathbf{I} - \mathbf{J}_{\mathbf{ci}}^{+}\mathbf{J}_{\mathbf{ci}}]$ and the allowable forces by $[\mathbf{J}_{\mathbf{ci}}^{+}\mathbf{J}_{\mathbf{ci}}]^{\mathbf{T}}$. These matrices are orthogonal projectors in joint space onto the allowable position and force variations respectively. A further explanation of this result is given in Chapter 5 of [8]. The projectors will be abbreviated to

$$ {}_j\mathbf{P}_{ap} = [\mathbf{I} - \mathbf{J}_{\mathbf{ci}}^{+}\mathbf{J}_{\mathbf{ci}}] \quad \text{and} \quad {}_j\mathbf{P}_{af} = [\mathbf{J}_{\mathbf{ci}}^{+}\mathbf{J}_{\mathbf{ci}}]^{\mathbf{T}} = [\mathbf{I} - ({}_{ap}^{j}\mathbf{P})^{\mathbf{T}}]. \tag{3.6} $$

The subscript j denotes joint space, and ap and af stand for allowable positions and allowable forces respectively. It can be observed that these projection matrices project onto the nullspace of the respective constraint directions. This can further be related to the concept of task priority, in which tasks with lower priority are performed in the null-space of higher priority tasks [28]. An important observation is that the mapping onto the allowable force and position spaces are in this method purely determined by the kinematics of the robot.

Multiple constraints

If there are several contact points, projection matrices are calculated for each constraint, and the final projection matrices are found by taking the union and intersect of the different ${}_j\mathbf{P}_{af}$ and ${}_j\mathbf{P}_{ap}$ respectively.

Passive joints

The presence of passive joints in the robot imposes additional constraints on the allowable forces. This is because the force in a passive joint is uncontrollable. [8] present two methods of including this constraint. One of which is using a diagonal matrix $\mathbf{A}$ that denotes which joints are passive and which are active. A 1 on the diagonal indicates an active joint whereas a 0 indicates a passive joint. This matrix has to be manually initialized before controlling the robot. It can then be combined with the allowable force projection by taking the

intersect of the space spanned by $\mathbf{A}$ and ${}_j\mathbf{P}_{af}$.

Task analysis

An end effector task may consist of both a movement and force application onto a surface. The projectors ${}_j\mathbf{P}_{af}$ and ${}_j\mathbf{P}_{ap}$ make sure the force and movement are performed in the allowable force and movement directions respectively. A specific task may however not be possible to perform within the restrictions of these spaces.

Essential position variables are defined as the movement directions of the end effector which must be controlled in order to perform the task correctly. At the same time, *arbitrary position variables* are those directions of movement which do not have to be controlled precisely. The terms *essential force variables* and *arbitrary force variables* follow the same logic, just for the force directions. According to [8], the total number of essential variables, position plus force, is equal to the minimum number of controllable actuators in the manipulator necessary for performing the task.

The essential position and force direction can be described by the matrices (3.7) and (3.8) respectively.

$$\mathbf{E}_p = \begin{bmatrix} \mathbf{e}_{p1} & \dots & \mathbf{e}_{p\alpha} \end{bmatrix} \tag{3.7}$$

$$\mathbf{E}_f = \begin{bmatrix} \mathbf{e}_{f1} & \dots & \mathbf{e}_{f\beta} \end{bmatrix} \tag{3.8}$$

Each column vector $\mathbf{e}_{fi}$, $\mathbf{e}_{pi}$ describes one direction. Following, the number of essential position directions is α and the number of essential force directions is

β. The orthogonal projections onto the essential position and force spaces are given by (3.9) and (3.10) respectively.

$$ {}_w\mathbf{P}_{ep} = \mathbf{E}_p(\mathbf{E}_p^T\mathbf{E}_p)^{-1}\mathbf{E}_p^T \tag{3.9} $$

$$ {}_w\mathbf{P}_{ef} = \mathbf{E}_f(\mathbf{E}_f^T\mathbf{E}_f)^{-1}\mathbf{E}_f^T \tag{3.10} $$

The inverse in the two equations above is, probably by mistake, not included in [8]. The w denotes that the projectors are defined in work space coordinates. In order to combine these projectors with the allowable position and force projectors, it is desirable to define them in the joint space coordinate system. The precondition for this to be possible for the essential position space is that the number of joints linking the base of the manipulator to the end effector is greater than or equal to three (given the two dimensional case). For the essential force space, the number of *active* joints has to be greater than or equal to three. The joint space projectors can then be found by

$$ {}_j\mathbf{P}_{ep} = \mathbf{J}_t^+ \, {}_w\mathbf{P}_{ep}\mathbf{J}_t \tag{3.11} $$

$$ {}_j\mathbf{P}_{ef} = \mathbf{J}_c^+ \, {}_w\mathbf{P}_{ef}(\mathbf{J}_c^T)^+. \tag{3.12} $$

The Jacobian $\mathbf{J}_t$ relates the task specific coordinates to the joint space. Section 3.1.2 explains how this Jacobian can be found for the snake robot case.

Eventually, the projections in joint space that project onto the allowable motion and force spaces *and* result in the desired motion and force necessary

to perform the task are given by (3.13) and (3.14) respectively.

$$ {}_j\mathbf{F}_p = {}_j\mathbf{P}_{ap}({}_j\mathbf{P}_{ep}\,{}_j\mathbf{P}_{ap})^+\,{}_j\mathbf{P}_{ep} \tag{3.13} $$

$$ {}_j\mathbf{F}_f = {}_j\mathbf{P}_{af}({}_j\mathbf{P}_{ef}\,{}_j\mathbf{P}_{af})^+\,{}_j\mathbf{P}_{ef} \tag{3.14} $$

3.1.2 Dynamic HPFC

The solution of West and Asada [8] does not take the manipulator dynamics into account. Nevertheless, in a real system, the dynamics play a significant role in the resulting behavior of the robot. For this reason, Yoshikawa [10] designed the dynamic hybrid control method which incorporates the constraints into the manipulator dynamics. More specifically, the solution of [8] filters the commanded joint torques and velocities to conform to the constraints and the essential variable space. This is explained in more detail in 3.1.1. The essence of the solution of [10] however, is that the robot dynamics and constraint equations are combined before the commanded torques and angles are calculated.

This section aims at describing the improved method, and the content is based on the paper of [10]. The symbolic conventions used are for simplicity the same as in the paper. The next section will explain further how the theory and these symbols apply to the snake robot case and the snake robot specific theory presented in the previous chapter (2.2-2.4).

It is worth noting that the solution of Yoshikawa is designed for a robot manipulator with a static base where the only constraint present is targeted at the manipulator end effector. For this reason, special effort has been put into

finding a suitable formulation of the snake robot constraints. Additionally, the difference between the coordinate spaces introduced in the paper are easy to confuse and special attention has been directed at thoroughly defining these spaces for the snake robot so that the following calculations can be as clear and logical as possible.

Description of variables

A brief overview of the most significant new variables used in this section is provided below. The most general variables correspond to the ones presented in 2.2.

- $\mathbf{r_t}$: task coordinates related to the contact point (task) coordinate system
- $\mathbf{r_c}$: constraint coordinates related to the fixed obstacle coordinate system
- $\mathbf{E}_F$: axes of constraints on the force in the task frame
- $\mathbf{E}_P$: axes of constraints on the position in the task frame
- $\mathbf{J}_t$: Jacobian relating the joint coordinates $\dot{\mathbf{q}}$ to the task coordinates $\dot{\mathbf{r}}_t$
- $\boldsymbol{\tau}_c$: control torque
- $\boldsymbol{\tau}_F$: torque from the desired force
- $\boldsymbol{\tau}_P$: torque from the desired position

The task and obstacle (constraint) frames $\mathbf{r_t}$ and $\mathbf{r_c}$ are visualized in Figure 3.1. The values for both $r_{t,i} = [x_{t,i}, y_{t,i}, \theta_{t,i}]$ and $r_{c,i} = [x_{c,i}, y_{c,i}, \theta_{c,i}]$ are defined with respect to the base frame. It should be noted that the task frame, as opposed to the constraint frame, is allowed to rotate with respect to the base frame.

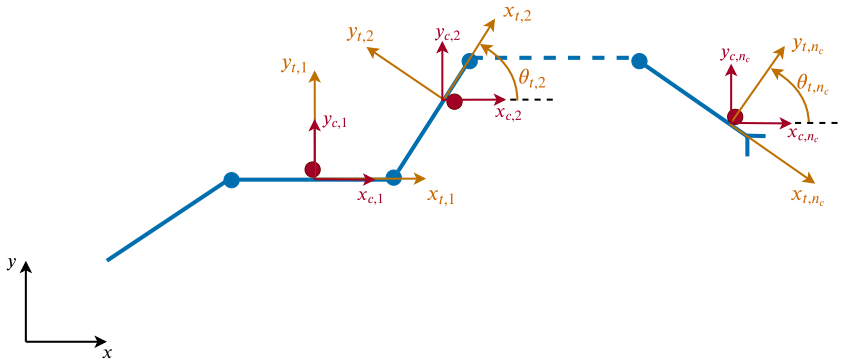


Figure 3.1: Model of snake robot and obstacles illustrating the task and contact frames

Description of constraints

In order to take the dynamics into account, the constraints are directly included into the dynamic equations of motion of the robot. This is done by expressing the constraints as a set of hypersurfaces that the robot can not physically pass. It should be noted that the focus in the paper of Yoshikawa [10] is on manipulator end-effector constraints and not general constraints, which should be used in the snake robot case. The constraint hypersurfaces are also expressed in the end-effector coordinates. Another important aspect of the paper is that it only addresses bilateral hypersurfaces (and not unilateral surfaces), meaning that the effector is prohibited from leaving the surface in any direction.

The m hypersurfaces expressing a given constraint are given by

$$p_i(\mathbf{r}_t) = 0, \quad i = 1, 2, ..., m, \tag{3.15}$$

where $\mathbf{r}_t$ is the end effector position in a fixed reference frame. Differentiating (3.15) with respect to time yields

$$\mathbf{E}_F \dot{\mathbf{r}}_t = 0, \tag{3.16}$$

where the vectors of $\mathbf{E}_F$ are the unit normal vectors to the hypersurfaces in (3.15).

By comparing the expression (2.24) of the constraint on link i found in 2.4 to (3.16), it is possible to extract the matrix $\mathbf{E}_{F,i}$ and a logical choice of $\mathbf{r}_{t,i}$ and $\dot{\mathbf{r}}_{t,i}$ presents itself. Specifically, if one chooses

$$\mathbf{r}_{t,i} = \begin{bmatrix} x_{t,i} & y_{t,i} & \theta_{t,i} \end{bmatrix}^T \in \mathbb{R}^3, \tag{3.17}$$

then $\dot{\mathbf{r}}_{t,i} = [\dot{x}_{t,i}, \dot{y}_{t,i}, \dot{\theta}_{t,i}]^T \in \mathbb{R}^3$ and the matrix $\mathbf{E}_{F,i}$ can by comparison be found as

$$\mathbf{E}_{F,i} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^3. \tag{3.18}$$

The subscript i will now be used as the constraint number, where n_c is the number of constraints. This is the same as the number of contact points. The angle $\theta_{t,i}$ of the link at the contact point is the angle θ of the link with respect to the base frame (see Figure 3.1). It can by inspection be seen that $\mathbf{E}_{F,i}$ fulfills the criteria of being of unit size.

The coordinate space $\mathbf{r}_t$ should be able to aid in expressing all the constraints present on the snake robot. Therefore, it is chosen as

$$\mathbf{r}_t = \begin{bmatrix} \mathbf{r}_{t,1}^T & \mathbf{r}_{t,2}^T & \dots & \mathbf{r}_{t,n_c}^T \end{bmatrix}^T \in \mathbb{R}^{3n_c}. \tag{3.19}$$

The same goes for $\dot{\mathbf{r}}_t$. Furthermore, the matrix $\mathbf{E}_F$ describing the unit normal vectors to all the hypersurfaces can now be written as

$$\mathbf{E}_F = \begin{bmatrix} \mathbf{E}_{F,1} & \mathbf{0}_{1\times 3} & \dots & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \ddots & & \vdots \\ \vdots & & \ddots & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \dots & \mathbf{0}_{1\times 3} & \mathbf{E}_{F,n_c} \end{bmatrix} \in \mathbb{R}^{n_c \times 3n_c} \tag{3.20}$$

Differentiating (3.16) further gives

$$\mathbf{E}_F \ddot{\mathbf{r}}_t + \mathbf{a}_{rF} = 0, \quad \mathbf{a}_{rF} = \dot{\mathbf{E}}_F \dot{\mathbf{r}}_t, \tag{3.21}$$

where the first term is believed to be the acceleration in the direction of the hypersurface, i.e. the force direction, and $\mathbf{a}_{rF} \in \mathbb{R}^{n_c}$ is believed to be an acceleration resulting from the relative movement of the task frame.

Furthermore, $\mathbf{E}_P$ is chosen so that all the vectors in the relation

$$\mathbf{E} = \begin{bmatrix} \mathbf{E}_P \\ \mathbf{E}_F \end{bmatrix} \tag{3.22}$$

are mutually independent unit vectors. The matrix $\mathbf{E}_F$ represents the coordinate axes normal to the constraint surfaces, and $\mathbf{E}_P$ represents the coordinate axes complementing $\mathbf{E}_F$. Another way to display this is seeing $\mathbf{E}_F$ and $\mathbf{E}_P$ as the axes for force and position constrained directions respectively.

From the equations

$$\mathbf{E}\dot{\mathbf{r}} = \begin{bmatrix} \mathbf{E}_P \dot{\mathbf{r}}_t \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{E}\ddot{\mathbf{r}}_t = \begin{bmatrix} \mathbf{E}_P \ddot{\mathbf{r}}_t \\ -\mathbf{a}_{rF} \end{bmatrix} \tag{3.23}$$

it can be seen that the velocity to the constraint surface is zero, which is natural seeing as the end-effector should be physically unable to move through the surface.

For the snake robot, a simple choice of $\mathbf{E}_{P,i}$ with unit vectors complementing $\mathbf{E}_{F,i}$ is by inspection found to be

$$\mathbf{E}_{P,i} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 3}. \tag{3.24}$$

This is a very logical choice, seeing as the snake robot link in contact is allowed to rotate about the obstacle and move along the obstacle. Again, $\mathbf{E}_{P,i}$ corresponds to the i'th constraint. Combining all $\mathbf{E}_{P,i}$ gives

$$\mathbf{E}_P = \begin{bmatrix} \mathbf{E}_{P,1} & \mathbf{0}_{2\times 3} & \dots & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 3} & \ddots & & \vdots \\ \vdots & & \ddots & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 3} & \dots & \mathbf{0}_{2\times 3} & \mathbf{E}_{P,n_c} \end{bmatrix} \in \mathbb{R}^{2n_c\times 3n_c} \tag{3.25}$$

and $\mathbf{E} \in \mathbb{R}^{3n_c \times 3n_c}$.

Kinematics and dynamics

In [10], the relation between the joint variable vector $\mathbf{q}$ and the end effector position $\mathbf{r}_t$ is expressed as

$$\mathbf{r}_t = \mathbf{c}(\mathbf{q}). \tag{3.26}$$

The following equations are generated by differentiating 3.26.

$$\dot{\mathbf{r}}_t = \mathbf{J}_t \dot{\mathbf{q}}, \quad \mathbf{J}_t = \frac{\partial \mathbf{c}(\mathbf{q})}{\partial \mathbf{q}^T} \tag{3.27}$$

$$\ddot{\mathbf{r}}_t = \mathbf{J}_t \ddot{\mathbf{q}} + \mathbf{a}_q, \quad \mathbf{a}_q = \dot{\mathbf{J}}_t \dot{\mathbf{q}} \tag{3.28}$$

For the snake robot case, the matrix $\mathbf{J}_t$ contains the Jacobians of all the contacts.

$$\mathbf{J}_t = \begin{bmatrix} \mathbf{J}_{t,1} \\ \mathbf{J}_{t,2} \\ \vdots \\ \mathbf{J}_{t,n_c} \end{bmatrix} \in \mathbb{R}^{3n_c \times N} \tag{3.29}$$

The Jacobian $\mathbf{J}_{t,i} \in \mathbb{R}^{3 \times N}$ for a single contact point with respect to the contact point task variable vector $\mathbf{r}_{t,i}$ is found as

$$\mathbf{J}_{t,i} = \begin{bmatrix} \frac{\partial x_{t,i}}{\partial q_1} & \cdots & \frac{\partial x_{t,i}}{\partial q_N} \\ \frac{\partial y_{t,i}}{\partial q_1} & \cdots & \frac{\partial y_{t,i}}{\partial q_N} \\ \frac{\partial \theta_{t,i}}{\partial q_1} & \cdots & \frac{\partial \theta_{t,i}}{\partial q_N} \end{bmatrix}. \tag{3.30}$$

Furthermore, $\mathbf{a}_q \in \mathbb{R}^{n_c}$.

Calculation of the joint driving force

The torque resulting from the contact force is found by

$$\boldsymbol{\tau}_F = \mathbf{J}_t^T \mathbf{E}_F^T \mathbf{f}_F. \tag{3.31}$$

The total torque $\boldsymbol{\tau}$ applied to the robot will be the difference between the motor torque $\boldsymbol{\tau}_c$ and the constraint torque $\boldsymbol{\tau}_F$.

$$\boldsymbol{\tau} = \boldsymbol{\tau}_c - \boldsymbol{\tau}_F \tag{3.32}$$

Combining the torque in (3.32) with the equations of motion given in (2.13) gives

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{J}_t^T \mathbf{E}_F^T \mathbf{f}_F = \boldsymbol{\tau}_c - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \tag{3.33}$$

and

$$\mathbf{E}_F \mathbf{J}_t \ddot{\mathbf{q}} = -\mathbf{E}_F \mathbf{a}_q - \mathbf{a}_{rF}. \tag{3.34}$$

Lastly, it can be shown that combining (3.33) and (3.34) yields the expressions

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{b}_1 + (\mathbf{E}_F\mathbf{J}_t)^T\mathbf{K}(\mathbf{b}_2 - \mathbf{E}_F\mathbf{J}_t\mathbf{M}^{-1}\mathbf{b}_1)), \tag{3.35}$$

$$\mathbf{f}_F = -\mathbf{K}(\mathbf{b}_2 - \mathbf{E}_F\mathbf{J}_t\mathbf{M}^{-1}\mathbf{b}_1). \tag{3.36}$$

$\mathbf{K}$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are given by

$$\begin{aligned} \mathbf{K} &= (\mathbf{E}_F\mathbf{J}_t\mathbf{M}^{-1}\mathbf{J}_t^T\mathbf{E}_F^T)^{-1} \\ \mathbf{b}_1 &= \boldsymbol{\tau}_c - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \\ \mathbf{b}_2 &= -\mathbf{E}_F\mathbf{a}_q - \mathbf{a}_{rF}. \end{aligned} \tag{3.37}$$

Eventually, it is possible to calculate the joint control torque. It consists of a component based on the desired movement $\ddot{\mathbf{r}}_{t,d}$ in the contact point frame and a component based on the desired force $\mathbf{f}_{Fd}$ applied to the constraint surfaces. The direction of the desired force should correspond to which side of the snake robot the obstacle is. This can either be implemented into the higher level controller requesting these forces, or one could introduce a vector γ like in 2.4. This vector consists of positive and negative units corresponding to the different obstacle placements. When multiplied with $\mathbf{f}_{Fd}$ it will simply negate the forces that should be applied to obstacles on the right side of the snake robot.

$$\boldsymbol{\tau}_c = \boldsymbol{\tau}_P + \boldsymbol{\tau}_F \tag{3.38}$$

The torque $\boldsymbol{\tau}_P$ is found by solving the equations of motion given in (2.13) based on the desired values of the joint accelerations.

$$\boldsymbol{\tau}_P = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_d + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \tag{3.39}$$

$$\boldsymbol{\tau}_F = \mathbf{J}_t^T \mathbf{E}_F^T \mathbf{f}_{Fd} \tag{3.40}$$

$$\ddot{\mathbf{q}}_d = \mathbf{J}_t^+ (\mathbf{E}^{-1} \begin{bmatrix} \ddot{\mathbf{r}}_{EP,t,d} \\ -\mathbf{a}_{rF} \end{bmatrix} - \mathbf{a}_q) + (\mathbf{I} - \mathbf{J}_t^+ \mathbf{J}_t)\mathbf{k} \tag{3.41}$$

Here the vector $\ddot{\mathbf{r}}_{EP,t,d} = \mathbf{E}_P \ddot{\mathbf{r}}_{t,d}$. Furthermore, $\mathbf{J}_t^+$ denotes the pseudo inverse of the Jacobian. The last term is included for redundant manipulators, where the vector $\mathbf{k}$ is an arbitrary time function representing the arbitrariness of the joint accelerations. It can be observed that the term $(\mathbf{I} - \mathbf{J}_t^+ \mathbf{J}_t)$ is the same as the projection matrix ${}_j\mathbf{P}_{ap}$ onto the allowable motion space from 3.1.1.

According to Yoshikawa [10] the position and force can be simultaneously controlled by applying the sum of the joint torque $\boldsymbol{\tau}_P$ for achieving the desired acceleration and the joint torque $\boldsymbol{\tau}_F$ for achieving the desired force as long as the robot is not in a singular state.

Describing the desired acceleration through $\ddot{\mathbf{r}}_{t,d}$ might not be the most intuitive task. It is therefore suggested that $\mathbf{E}_P$ is chosen in such a manner that a function $\dot{\mathbf{r}}_{P,t} = s(\mathbf{r}_t)$ exists and satisfies

$$\dot{\mathbf{r}}_{P,t} = \mathbf{E}_P \dot{\mathbf{r}}_t. \tag{3.42}$$

If this holds, then

$$\ddot{\mathbf{r}}_{P,t} = \ddot{\mathbf{r}}_{EP,t} + \mathbf{a}_{rP}, \quad \mathbf{a}_{rP} = \dot{\mathbf{E}}_P \dot{\mathbf{r}}_{P,t}. \tag{3.43}$$

Given that $\mathbf{r}_{P,t}$ is given by $\mathbf{r}_{P,t,d}$ and the described dynamics and constraints are correct, the system given by (3.38)-(3.41) will be able to realize both the desired position and force.

3.2 Passive joints consideration

The problem that arises when computing $\boldsymbol{\tau}_P$ from (3.39) in 3.1.2, is that the motor torques will be calculated for all joints, including the passive joints. Since the passive joints are unactuated, it is not feasible to pursue all parts of this control command. The objective is thus to change the dynamic calculations to only apply torques to the active joints. It should be noted that this challenge is more crucial for snake robots with few (active) joints. This is because a large number of active joints will dominate the control and a bigger part of the control torque can be realised.

One idea is using the dynamic coupling characteristics between the passive and active joints, as proposed by Arai et al. [29]. This method assumes that there is a certain amount of active joints that can take arbitrary values determined by the other active and passive joints desired to control. That means that all joint variables still cannot be controlled to their desired values and it has to be determined which joints this should apply to. The reason for this is that the system is underactuated. A similar approach is suggested by Transeth et al. [30]. However this method assumes that only the active joints

make out the control reference. Because it is not yet clear whether or not it will be desired to control the value of the passive joints in the HOAL scheme, the rest of this section is based on the work of Arai et al. [29].

The total number of joints is now denoted n, whereas r of these joints are active. To start off, the elements of the familiar variables $\mathbf{q}$ and $\boldsymbol{\tau}$ should be rearranged with dimensions as follows

$$\mathbf{q} = \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{\psi} \end{bmatrix} \begin{matrix} n-r \\ r \end{matrix} = \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{\psi}_{act} \\ \boldsymbol{\psi}_{pas} \end{bmatrix} \begin{matrix} n-r \\ 2r-n \\ n-r \end{matrix} \tag{3.44a}$$

$$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\tau}_{act} \\ \mathbf{0} \end{bmatrix} \begin{matrix} r \\ n-r \end{matrix}. \tag{3.44b}$$

Here $\boldsymbol{\psi}$ are the r joint variables, both passive and active, that will be controlled to their desired value. $\boldsymbol{\phi}$ contains the remaining $n-r$ active joint values. Furthermore, the corresponding generalized force vector $\boldsymbol{\tau}$ is divided into an active and a passive part. The part corresponding to the passive joints is zero because the generalized forces on the passive joints by definition are zero.

The inertia and Coriolis matrices $\mathbf{M}(\mathbf{q})$ and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ can now be rearranged accordingly. The dimensions of the rows and columns are indicated in (3.45a) and (3.45b).

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} \mathbf{M}_{11}(\mathbf{q}) & \mathbf{M}_{12}(\mathbf{q}) \\ \mathbf{M}_{21}(\mathbf{q}) & \mathbf{M}_{22}(\mathbf{q}) \\ n-r & r \end{bmatrix} \begin{matrix} r \\ n-r \end{matrix} \tag{3.45a}$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} \mathbf{C}_1(\mathbf{q}, \dot{\mathbf{q}}) & r \\ \mathbf{C}_2(\mathbf{q}, \dot{\mathbf{q}}) & n-r \end{bmatrix} \tag{3.45b}$$

Using (3.45a) and (3.45b), the equations of motion (2.13) can be split into the two separate equations

$$\mathbf{M}_{11}\ddot{\boldsymbol{\phi}} + \mathbf{M}_{12}\ddot{\boldsymbol{\psi}} + \mathbf{C}_1 = \boldsymbol{\tau}_{act} \tag{3.46a}$$

$$\mathbf{M}_{21}\ddot{\boldsymbol{\phi}} + \mathbf{M}_{22}\ddot{\boldsymbol{\psi}} + \mathbf{C}_2 = \mathbf{0}. \tag{3.46b}$$

When the values for $\mathbf{q}$ and $\dot{\mathbf{q}}$ are inserted and the desired values $\ddot{\boldsymbol{\psi}}_d$ are assigned to the accelerations $\ddot{\boldsymbol{\psi}}$, (3.46b) is considered a linear equation with regard to $\ddot{\boldsymbol{\phi}}$. The coefficient matrix $\mathbf{M}_{21}$ corresponds to the dynamic coupling between the accelerations of $\boldsymbol{\phi}$ and the generalized forces of the passive joints, and it depends on the structure and mass distribution of the manipulator [29].

Solving (3.46b) for $\ddot{\boldsymbol{\phi}}$ yields

$$\ddot{\boldsymbol{\phi}} = -\mathbf{M}_{21}^{-1}\mathbf{M}_{22}\ddot{\boldsymbol{\psi}}_d - \mathbf{M}_{21}^{-1}\mathbf{C}_2. \tag{3.47}$$

This is given that the matrix $\mathbf{M}_{21}$ is nonsingular. Arai et al. [29] proved that this is also the condition for output controllability. The generalized forces on the active joints can be found by substituting (3.47) into (3.46a).

$$\boldsymbol{\tau}_{act} = (\mathbf{M}_{12} - \mathbf{M}_{11}\mathbf{M}_{21}^{-1}\mathbf{M}_{22})\ddot{\boldsymbol{\psi}}_d + \mathbf{C}_1 - \mathbf{M}_{11}\mathbf{M}_{21}^{-1}\mathbf{C}_2 \tag{3.48}$$

If the generalized forces $\boldsymbol{\tau}_{act}$ are applied to the active joints, the resulting

accelerations will be $\ddot{\boldsymbol{\phi}}$ and $\ddot{\boldsymbol{\psi}}_d$ [29].

From the dimensions of $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ it can be deduced that if the number of active joints r equals the total number of joints n, all joint variables can be controlled. This is what would be referred to as a fully actuated system. If, on the other hand, there were no active joints, the dimension of $\boldsymbol{\psi}$ would be zero and all variables would be uncontrollable. These two cases are both quite intuitive. The more vital question is exactly how many of the joints are required to be active in order to achieve the desired control goal. This is equivalent to the question of how many of the joint variables are desired to be controlled, and is dependent on the full HOAL algorithm.

Seeing as the joints desired to be controlled will change continuously for a snake robot traversing a terrain with obstacles, this is probably not the most robust solution. The bottom line is however that the control torques are mapped to the active joints and all of them can now be commanded to the snake robot. The method is tested in 5.5 and further discussed in 6.1.3.

3.3 The utility of dynamic HPFC in snake robot locomotion

Most of the snake robot locomotion strategies today are intended for flat surfaces. Lateral undulation and sidewinding are two of these strategies presented in 2.1.1. Even though they are based on flat surfaces, they require some amount of ground friction to yield propulsion. Either way, the snake robot body motions are the basis for the propulsion. A position controller for the joint angles of the snake robot would thus induce a satisfying behavior.

Consequently, a hybrid position/force controller would simply be an overkill for these strategies. The concertina locomotion strategy, on the other hand, is conducted in narrow spaces where the snake robot anchors itself against the environment. This is also explained further in 2.1.1. A joint controller is sufficient in this case as well if only the curling up and stretching out of the robot is considered. It might however be desired for the snake robot to push against its environment at the same time as this movement takes place. A hybrid position/force controller could then be useful to simultaneously realize these control goals.

Compliance control, presented in 2.1.4, is another control option for adapting to the environment. The main difference between compliance control and dynamic HPFC is that the compliance controller is a so called reactive controller, meaning that the robot will change its curvature according to the environment when a contact feedback has been received. The dynamic HPFC contrarily uses a feedforward term for dynamic force, enabling it to always control both the force and position precisely. The success of this method does to a large degree depend on a correct modeling of the dynamics of the robot and good perception of its environment.

The locomotion method which can take the most advantage of the dynamic HPFC is the OAL method presented in 2.1.3. The possibility to control position and force simultaneously and independently in different directions allows the snake robot to both move between obstacles and push against them at the same time. Furthermore, the dynamical control facilitates a dynamical or compliant, rather than stiff, behavior. This is beneficial for the mechanical parts of the snake robot which can then experience a lower level of strain and wear and tear.

3.4 Application challenges related to dynamic HPFC

There are surely several ways of implementing and designing the dynamic HPFC control logic on a snake robot. This section will focus on the challenges related to the application of the suggested design scheme in 3.1.2. Special attention is paid to the chosen variable spaces and the consequences of configuration transitions when the snake robot moves.

3.4.1 Computational challenges

Whenever the snake robot achieves successful forward or backward propulsion it will slide along the obstacles that are by its side. Eventually the contact between a given link and an obstacle will be lost. At this point, the obstacle will either be left alone or come in contact with a neighboring link. These two scenarios are illustrated in Figures 3.2 and 3.3 respectively. If the contact is completely lost, it means that one constraint is lost as well and $n_c = n_c - 1$. This will in turn lead to the variable $\mathbf{r}_t$, describing the position of the contacts, shrinking. Consequently the mapping matrix $\mathbf{E}$ to the allowed force and movement directions will also shrink. The most significant challenge in this case is changing the dimensions of the affected variables in real-time. One option could be keeping all variable spaces unchanged and instead set the parts of $\mathbf{E}$ corresponding to the lost constraint to zero so that it has no impact on the following solution.

If, on the other hand, the snake simply slides in a way that the contact is transferred to an adjoining link, the size n_c and dimensions of all variables will remain unchanged. In addition to this, the obstacle will lie on the same side of the snake as it already was, which enables the direction of the desired force

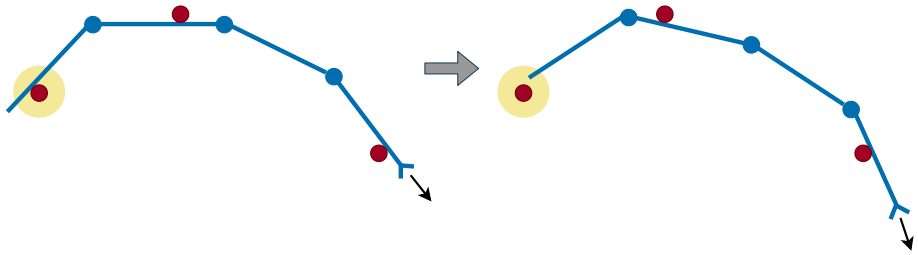


Figure 3.2: Snake robot losing contact with obstacle

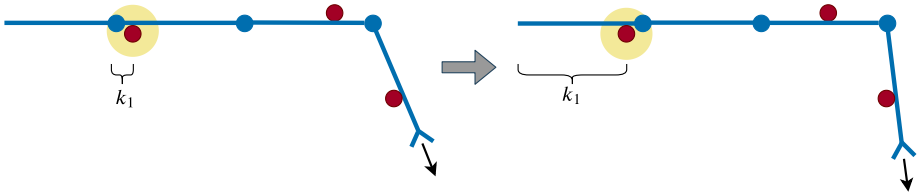


Figure 3.3: Obstacle changing contact from one link to another

application to the obstacle to stay the same. Thus, the constraint will generally be on the same form. The only thing that has to change is the description of the position variables in $\mathbf{r}_t$ belonging to the new contact point. Again, the change is most probably slim since the two first variables (x_t, y_t) describe the position of the contact point, and the obstacle is assumed to be static (meaning its position will not change even though the calculation of the position changes). Furthermore, the adjoining link adopting the contact is probable to have a similar angle θ_t relative to the base frame as the previous link given that the desired path is well designed and defined. From this it follows that the joint variable ϕ_c back to the obstacle frame will stay similar to what it was as well. These aspects obviously make the implementation simpler and the control

sequence smoother and more predictable.

Even though the numerical value of the mentioned variables do not change drastically, the Jacobian will have to be recalculated since the variables now are described by a new subset of the joint variables. Finding a new expression for the Jacobian matrix and its derivative in real-time is not a trivial, nor fast, operation. One option would of course be pre-computing $\mathbf{J}$ and $\dot{\mathbf{J}}$ for every link that could be in contact with an obstacle and only employ the ones that are relevant at the specific time instance. There is however still a challenge related to this case. The generalized joint variables $\mathbf{q}$ contain the distances k from the contact points to the preceding joints. It is important that these distances are not mixed and that they are only used once. In other words, two contact points should not be described by the same k. As a consequence of this, the Jacobian for every link would have to be computed with all the possible k's. Logically, this would in turn lead to a large number of pre-computed matrices as the number of links and obstacles grow. In addition to this, the right set of Jacobians would have to be chosen in real-time while administering that they all use unique k's. This should be done without changing the existing setup more than necessary in order to avoid jumps in the control. It is very much an achievable task, but at the same time an extra challenge.

With a contact moving from one link to another, the corresponding joint variable k describing the position of the contact point will change and the manner in which it is computed will also change. This is once again an achievable, yet challenging task to perform in real-time. It should also be noted that the value of this k will probably experience a jump. This is because the contact moves from the end of a link to the beginning of a link, or vice versa, and the distance is always measured with respect to the preceding joint of

the link in contact. This is illustrated in Figure 3.3. On the other hand, it is reassuring that the matrices $\mathbf{M}(\mathbf{q})$ and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ are unaltered by the change of a k. This is logical since these matrices describe the dynamics of the snake robot alone.

The biggest challenge arises if the number of contact points increase. This means that the dimension of all variables will have to increase correspondingly and the Jacobians have to be either re-computed or re-assembled. It is in this case important to keep in mind the challenge associated with memory allocation in the software being used.

3.4.2 Differences with the traditional manipulator case

One significant challenge related to snake robots as opposed to traditional robot manipulators is the presence of passive joints at the base of the robot. The passive joints of the snake robot that are attached to the base frame, $[\phi_0, x_0, y_0]$ are included in the description of the position of every link and contact point. This means that if they were actuated, they could have influenced all the points that are desired to control, given that the robot is not stuck between obstacles. The dynamic hybrid position/force controller could therefore introduce a desired joint acceleration value $\ddot{\mathbf{q}}_d$ for these passive joints when recalculating from the desired contact point accelerations $\ddot{\mathbf{r}}_{t,d}$. Since these joints are passive, they are also uncontrollable and a desired control input on the joints is not realizable. A greater number of joints will generally make this challenge less significant because it leads to a larger degree of controllability of all contact points.

When it comes to the assignments of traditional robot manipulators and

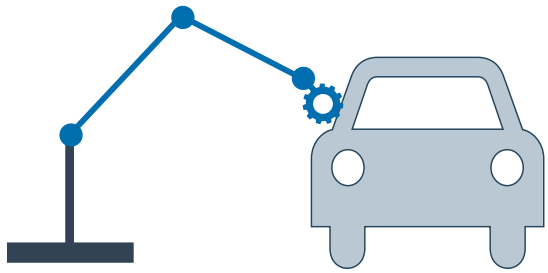


Figure 3.4: Industrial robot manipulator polishing a car

snake robots, they are usually a lot more repetitive and pre-defined for robot manipulators. A typical task for an industrial hybrid position/force controlled robot manipulator is polishing a known surface with a given pressure. An example is given in Figure 3.4. The snake robot might also be familiar with the shape of its current environment, but this is something that is constantly changing as the robot moves through the world. Thus, the position and force application requirements are constantly changing as well. Furthermore, since the snake robot is eventually meant to traverse outdoor environments, it will encounter different kinds of obstacles as illustrated in Figure 3.5. They can differ in both size and texture, and be either soft or rigid. Thus, there are a lot of factors the snake robot will have to take into account. For this project however, the environment and control goals are significantly simplified.

There is also a considerable difference in the typical constraints on a snake robot and a traditional manipulator. Manipulators usually only have constraints or obstacles in the close environment of the end effector in which the manipulator is performing a task. It might also have some limitations regarding the motion span of its joints, but this applies to snake robots as well, though

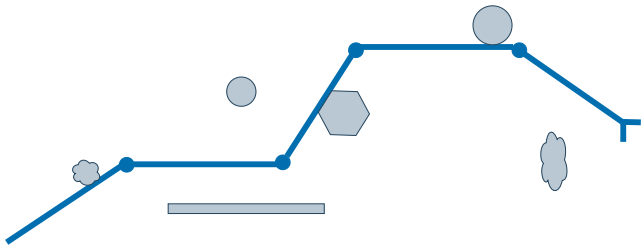


Figure 3.5: Snake robot in environment with varying obstacle types

perhaps to a smaller degree. The environment the snake robot is traversing may as mentioned contain various and spread out obstacles, making the constraints different from time to time. If, however, a robot is in an undesirable position between obstacles, a snake robot might have an advantage in that it can exit its current configuration from several different directions since no part of it is fixed to any point in the world.

3.5 Task analysis

Even though the control solution presented in 3.1.2 decomposes the workspace in force and movement directions, there are some restrictions as to which tasks can be performed by the robot. To analyze this, it is beneficial to look at what kind of tasks a snake robot is required to achieve. In particular, the tasks of a snake robot moving according to the OAL principle will be covered in this section. Both a higher level path following goal and a lower level control goal is presented. It is the lower level goal that will be further investigated in this project, but it is still important to keep in mind what the final purpose of the snake robot is.

Lastly, the task restrictions are discussed, followed by a mathematical adaption of the control scheme in 3.1.2 based on the theory of West and Asada [8] described in 3.1.1. This adaption is constructed to allow for the control of the task-relevant variables in cases where all task space variables cannot be controlled independently.

3.5.1 The overall task of the snake robot

The motivation behind implementing the OAL principle on a snake robot is to make it move from one point in space to another by utilizing the obstacles present in its environment. In a "real world" situation, solely moving around is not enough and the snake robot will normally have some auxiliary assignment as well. This can typically be documenting its surroundings with a camera or similar equipment attached to its foremost link, referred to as its head. This again makes the snake robot's exact path from the starting point to the end point relevant, and the path following of the head of the snake robot is thus the overall goal. When a path has been designed, this goal can be decomposed in tasks consisting of the global position and orientation of the snake robot head. In traditional robot manipulator theory, this would be referred to as the end-effector movement.

3.5.2 Lower level control tasks

Because the base of traditional manipulators are fixed to the ground, they can move the end effector in a desired manner simply by following the inverse kinematics. This is of course given that the robot's degrees of freedom satisfy the desired end effector movement. For snake robots, on the other hand, it is

more complicated. In order to reach the higher level path following goal, the snake robot has to push itself forward in a purposeful manner utilizing the obstacles in the environment. By purposeful it is meant that the direction of the force application against the obstacles have to conform with the desired propulsion direction given by the path. This is where the hybrid position/force control comes into play. The robot has to both position itself in a certain manner alongside the obstacles and push against them with a given force magnitude.

The general idea is that at any point in time a task will be given by a higher level path following algorithm which is assumed implemented. The information provided should include which obstacles are to be utilized and how the snake robot is to utilize them. In other words, the tasks sent to the hybrid position/force controller simply consist of a desired positioning (orientation) by every obstacle and force to be applied to the obstacles.

3.5.3 Task restrictions

The question is, what are the restrictions to which tasks the higher level path following algorithm can command the snake robot to perform? It is obvious that not any given combination of position and force can be simultaneously achieved.

The restrictions lie in the composition of the snake robot, meaning how many joints the robot consists of. A higher number of joints, and thus actuators, enables the robot to control a higher number of variables. At the same time, a higher number of contact points imposes a higher number of constraints on the system and therefore limits the controllability. That is not to say that few contact points are desired, because they are after all a necessity for successful

propulsion.

For every contact point there are three possible variables that can be controlled. In 3.1.2 it is defined that two of these variables belong to the position control. That is the movement along the obstacle and rotation by or around the obstacle. The remaining variable is reserved for the force against the obstacle.

To analyze the restrictions to how many and which of these variables can be controlled for a given robot, it is useful to look at the robot's closed kinematic chains (CKCs). A kinematic chain is said to be closed when it contains one or more loops [12]. In the snake robot case loops arise between points fixed in the world (base frame). These points are the base of the robot, meaning the start of the first virtual translational joint, and every obstacle contact point. This is always true since it is assumed that the position of the obstacles are fixed in the world.

The number of CKCs is the same as the number of contact points on the robot. However, they can be expressed in several different ways. The first method is to define all closed kinematic chains from the base to every contact point. This is depicted in Figure 3.6, where b denotes the base and $r_{c,i}$ denotes the different contact points.

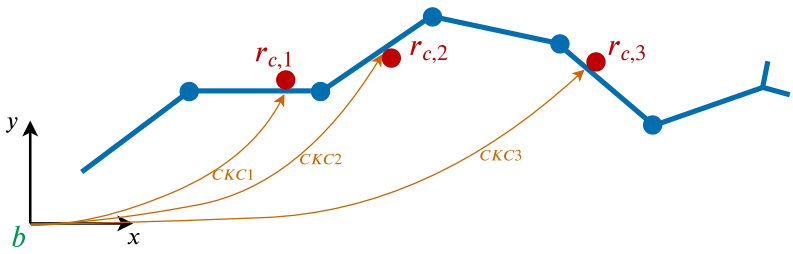


Figure 3.6: First method of defining the closed kinematic chains

It is also possible to define the CKCs sequentially through the robot. Starting at the base again, the first CKC will be from the base to the first contact point. Moving further, the second chain will be from the first to the second contact point, then from the second to the third, etc. This definition of the CKCs is depicted in Figure 3.7, and will from now on be referred to as a minimal CKC representation. There are still many possible ways of defining the CKCs, but the two mentioned methods are the most relevant in this case.

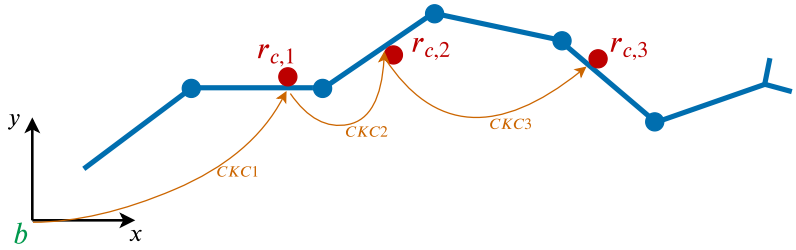


Figure 3.7: Second method of defining the closed kinematic chains

The number of available actuators for controlling the variables at a single contact point are limited by the placement of the preceding contact point. This is because the CKC between these two points always is the smallest CKC with the least number of actuators and thus the so called bottleneck for the control.

To better explain this, the example snake robot illustrated in figures 3.6 and 3.7 is studied. In Figure 3.6 there are four actuators included in the third CKC (CKC3) which is describing the third contact point. However, when looking at the closed kinematic chain for the same contact point described from its preceding contact point (depicted in Figure 3.7), only two actuators are included. Simply by studying the two figures it is also possible to see that the latter method always gives the minimal CKCs.

The ideal way of designing snake robots that require a high level of controllability is to include a very high number of joints. This way, even the minimal CKCs will have a sufficiently high number of actuators to make control at every contact point practically independent of the other contact points. Manipulators with such a large number of actuators are referred to as *hyperredundant manipulators*. According to [28], a manipulator is considered to be hyperredundant if its controllable configuration space degrees of freedom are comparable to, or exceed, its task space degrees of freedom. Furthermore, it is stated that such manipulators have an enhanced potential to use their extra joints for maneuvering within tight obstacle fields. The term hyperredundant can be adapted to snake robots as well, and it is now obvious that hyperredundancy is a requirement for an ideal OAL-driven snake robot.

Singularity avoidance

A common problem when using Jacobian matrices for the inverse kinematic computations is the presence of singularities. A robot configuration $\mathbf{q}$ is singular if the task Jacobian matrix $\mathbf{J}_t$ is rank-deficient here [28]. This configuration is not ideal since it means that it is impossible to generate task space velocities in certain directions. Task space velocities are for the snake robot velocities of the contact points vector $\mathbf{r}_t$. This is referred to as the end effector velocity in traditional robot manipulator theory.

An effect of a singularity is that the resulting joint space velocities can grow uncontrollably big. This effect is experienced not only at the singular configuration itself but also in its neighborhood [28]. According to Chiaverini et al. [28], the distance to these singularities can be characterized through measurements based on the determinant of the Jacobian or the individual

singular values of the matrix like the smallest singular value. If the snake robot consists of more joints than it needs for execution of the desired task, the additional degrees of freedom can be used to steer the snake robot away from singularities or avoid them at a planner level by analyzing these measurements. A more detailed description of the matter is presented in [28]. It should be noted that whether or not the robot is redundant depends both on the number of joints *and* the number of contact points desired to control, i.e. the task.

Furthermore, the snake robot does not necessarily lose mobility in all task directions even though it is at a singular configuration. This is noteworthy since it might not be desirable for the snake robot to move in every possible direction anyway. In 3.5.4 it is explained how the desirable or essential movement directions can be expressed and further utilized for control.

3.5.4 Task oriented control scheme

In 3.1.1 it is stated that there are certain directions of force and movement that are essential for performing a task. It is further claimed that the number of essential directions can not be greater than the number of controllable actuators in the robot necessary for performing the task. In 3.5.3 it is explained that the number of available controllable actuators is limited by the minimal CKC.

For the snake robot, the force and movement directions of a task are described by $\mathbf{E}_F$ and $\mathbf{E}_P$. The number of essential directions in $\mathbf{E}_F$ and $\mathbf{E}_P$ together can not exceed the number of active joints in the minimal CKC. In the example in Figure 3.7, a maximum number of three variables at the second contact point could be desired to control. However, it is seen that the corresponding CKC only contains one actuator, meaning that at most one

variable has to be chosen for control. In other words, the maximum number of essential variables that can be defined for performing a task at the second contact point is one.

As explained in 3.1.1, the essential position and force directions can be described by (3.9) and (3.10). The filters (3.13) and (3.14) can then be used to focus the control on these essential directions, also taking into account which directions the robot is allowed to move and apply force in. This is of course given that the requirements regarding the number of active joints are fulfilled.

The force filter ${}_j\mathbf{F}_f$, which is based on the allowable and essential force directions, can intuitively be combined with the input torque $\boldsymbol{\tau}_F$ for the desired force found by (3.40). It should be noted that the defined essential force directions and desired forces $\mathbf{f}_{F,d}$ have to correspond with each other. Combining the position filter ${}_j\mathbf{F}_p$ with the position control torque from (3.39) is unfortunately not as intuitive. This is because the filter is designed to map the joint velocities and not the joint torques. The dynamic HPFC control structure simply takes the desired contact point variables and directly computes the desired joint accelerations. This means that there is no intermediate step with joint velocities, as would have been convenient for the position filtering.

Under certain conditions and requirements, however, there are ways of combining the filter directly with the position control torque $\boldsymbol{\tau}_P$. First, it is known that as long as the position control is in fact performed in the position space, any acceleration $\ddot{\mathbf{q}}$ should lead to a change in velocity rather than force application. This again means there is a direct relationship between the joint accelerations and the velocities leading to the same "rules" being applicable for the two variables. Second, it is known that at very low velocities, the relationship between the joint torques and joint accelerations become

approximately linear. This is because the moment of inertia of the links is constant in the used snake robot model. Again, it can be stated that the same rules apply for the position torque and desired position joint accelerations. Conclusively, the position filter ${}_j\mathbf{F}_p$ can be used directly on the input torque for the desired position. It is again important that the essential position directions and desired position variables correspond.

Under the mentioned conditions, the new input control torques can be found by

$$\boldsymbol{\tau}_c = {}_j\mathbf{F}_p\boldsymbol{\tau}_P + {}_j\mathbf{F}_f\boldsymbol{\tau}_F. \tag{3.49}$$

3.6 Hybrid obstacle aided locomotion (HOAL)

The motivation behind using dynamic HPFC for the OAL is that this locomotion method requires both interaction with the environment, meaning force application, and purposeful movement of the snake robot body. Instead of constantly switching between force and position control of the snake robot, the dynamic HPFC can manage to control both of these attributes simultaneously in different directions. Furthermore, the dynamic method is studied in this thesis because it is believed that the dynamics play an important role in the behavior of the snake robot and contribute to a dynamic rather than stiff control. This is even more significant in outdoor environments with friction, as opposed to a sterile, frictionless simulation environment. The downside is that modeling the friction and other traits of the environment becomes harder as the surroundings get more complex and diverse.

Being able to predict the dynamical response of the snake robot enables the feed forward control and thus a smoother control trajectory. The dynamics of going from no contact to contact have not been modeled in this project, and is believed to be a very complex task. It is yet to be assessed whether or not the collision-modeling can profit the overall control of the snake robot at all.

This section does not give any definite answers to how the HOAL scheme can be implemented. It does however discuss related challenges and provides some suggestions as to how the problem can be approached.

3.6.1 General strategy for HOAL

There is much more to HOAL than just pushing against obstacles and changing the shape of the robot at the same time. The snake robot needs an end goal, or at least a temporary goal to follow. The approach for reaching this goal should be assisted by a defined path from the current location of the snake robot. This path should be designed not only to allow the snake robot to pass by constructive obstacles for propulsion, but also to make it propel itself forward in the best way possible from start to finish. The best way is here considered to be a path that is both minimizing the distance and the energy usage. These attributes are of course not independent of each other.

A suggestion to a general procedure for achieving HOAL is given below

1. Establish the position of the end goal of the snake robot head.

2. Design a path from the current location of the snake robot to the end goal. Here, the distance should be minimized while taking into account that enough obstacles have to be passed by to achieve propulsion. The

orientation of the robot along these obstacles should be taken into account as well.

3. Determine the desired force magnitudes against each obstacle based on the desired propulsion speed of the snake robot head.

4. Use dynamic HPFC and a suitable control structure to realize the desired forces and position/orientation defined by the steps above.

3.6.2 Conditions for propulsion

There are some criteria that have to be fulfilled for the HOAL algorithm to be successful. First, it requires knowledge of the environment and the position of the obstacles within a given relevant range. Second, as mentioned earlier, the modeling of the dynamics should be as accurate as possible.

It is also relevant to look at the position and force spaces of the robot. In which direction is the snake robot actually able to apply forces? In which directions can it move freely? In which directions will movement lead to forward propulsion? In what configurations will the snake robot be jammed? These are all very important questions that should be addressed in the construction of the desired path.

According to Stavdahl [1], the position or motion space $\mathbb{M}$ can be decomposed into a shape space $\mathbb{S}$ and a propulsion space $\mathbb{P}$. Furthermore, the force space $\mathbb{F}$ is referred to as the constraint space $\mathbb{C}$ because it does not directly yield propulsion. The knowledge about these spaces can be utilized to design a path and a set of desired forces that support the snake robot in propelling itself forward.

In 3.1.1 it is shown how the allowable motion and force spaces can be deduced from the constraint Jacobian $\mathbf{J}_c$. Based on the same logic, the Jacobian $\mathbf{J}_P$ relating the joint velocities to the desired velocity of the snake robot head along the given path can be used to find the propulsion space. The pseudoinverse $\mathbf{J}_P^+$ can be used to find the joint velocities related to a given snake robot head velocity $\mathbf{v}_P$. The relation is given in (3.50).

$$\dot{\mathbf{q}} = \mathbf{J}_P^+ \mathbf{v}_P \tag{3.50}$$

Furthermore, $(\mathbf{I} - \mathbf{J}_P^+ \mathbf{J}_P)$ represents the orthogonal projection matrix in the null space of $\mathbf{J}_P$ [28]. That means that joint velocities projected onto this space yield zero velocity for the snake robot head. The shape space S can now be found as the remaining part of the motion space $\mathbb{M}$ that is not spanned by $\mathbb{P}$.

For propulsion it is not desired that the snake robot head is moved in an arbitrary direction, but rather in a specific direction defined by the desired path. Therefore, a necessary condition for propulsion will be that this particular direction is within the span of the propulsion space. Finding the general criteria for when this is satisfied still remains to be answered.

According to Bayraktaroglu [20] a necessary condition for propulsion in a push-point-based locomotion approach is that the snake robot is in contact with and can push against at least three obstacles simultaneously. Bayraktaroglu [20] further states that the total exterior forces and moments applied to a system must overcome the inertia and compensate perturbations in order to make it move with respect to a fixed reference frame. From this it follows that the directions in which the snake robot pushes against the obstacles is vital. To visualize this, two examples are presented. The first one is given in Figure

3.8, in which it is obvious that no forward motion can be obtained from the resulting contact forces and the configuration is thus in the constraint space $\mathbb{C}$.

In Figure 3.9 however, one of the links is able to push against an obstacle with a force f_3 that has a component in the forward (right) direction, enabling the snake robot to slide along the obstacle. This means that the propulsion space $\mathbb{P}$ is non-empty. The two other forces f_1 and f_2 contribute to keeping the end part of the snake in line. Additionally, they can be regarded as a kind of base for the snake robot to push against. If they were not included, the attempt to push against the third obstacle would only lead to a change of the snake robot's shape, meaning the configuration would be in the shape space $\mathbb{S}$.

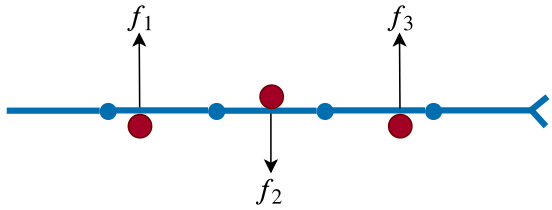


Figure 3.8: Force application yielding no propulsion

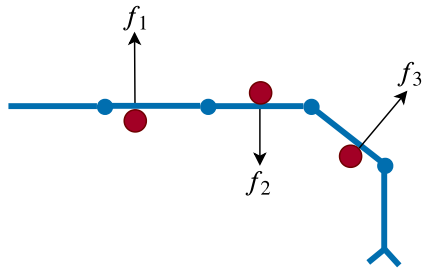
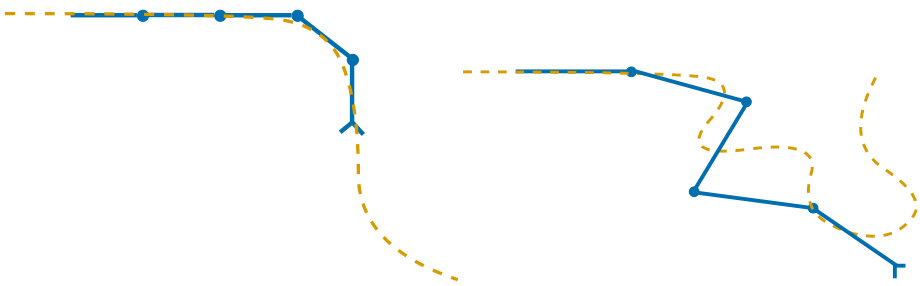


Figure 3.9: Force application yielding propulsion

The last example presenting forward propulsion does not include a desired trajectory for the snake robot to follow. Certain criteria will have to be met in the generation of the trajectory as well, and will typically be based on the composition of the snake robot. If the links are very long the snake robot will naturally not be able to shape itself along a very tightly curved path. If, on the other hand, the link length approaches zero the snake robot will resemble a natural snake and the limitations to its shape will decrease significantly. This type of robot is also referred to as a continuum robot, which according to Robinson et al. [31] is a continuously bending, infinite-degree-of-freedom structure. The goal in HOAL is not to adapt the snake robot composition to fit the path, but rather the path to fit the snake robot.

Figure 3.10 shows one case in which the curves of the desired path are sufficiently large for the snake robot to follow and one case in which the snake robot is physically unable to adjust itself perfectly to the desired path. The obstacles necessary for propulsion are left out in the illustrations.



(a) Successful path adjustment (b) Unsuccessful path adjustment

Figure 3.10: Snake robot desired path following

It is clear that a distinct set of conditions for the desired path should be

determined before the path planner is implemented. One hypothesis suggested by Stavdahl [1] is that if the path consists of straight lines and circle segments, the radius of the circle segments have to be at least the same length as the snake robot link. The resulting curvature can be defined through the osculating circle, which is the circle that best approximates the curve at a point [32]. The radius of this circle is denoted r, and the upper limit for the curvature is given by

$$\kappa_{max} \leq \frac{1}{r_{min}} = \frac{1}{L}. \tag{3.51}$$

Chapter 4

Simulator

The SnakeSIM simulator [33] has been chosen for performing the experiments considered relevant for this project. This chapter will give an overview of the software and how it is used in this project with the intention of making further contributions and usage easier. The simulator is based in the Robotic Operating System (ROS) [34] and Gazebo [35], and for an in-depth understanding, it is suggested to look at the ROS and Gazebo tutorials online.

4.1 Background info

SnakeSIM is a virtual rapid-prototyping framework developed by Sanfilippo et al. [33] to allow researchers to design and simulate *perception-driven obstacle-aided locomotion* (POAL) in a safe and rapid manner. The core of the simulator is based in the Robotic Operating System (ROS) [34]. This is where the robot model and control is defined. Furthermore, it is connected to Gazebo [35],

providing both a robust physics engine and a real-time graphical user interface. SnakeSIM has an additional interface towards the RViz visualization tool. This part will however not be explained further as it has not been applied in this project. Lastly, it should be noted that ROS is currently only supported on the Ubuntu operating system, and has a command line based interface. Ubuntu version 16.04, ROS Kinetic and Gazebo version 7.0 are used in this project.

The simulator has been modified to fit even better to the project. The main contribution is the implementation of the dynamic controller. Certain properties of the simulator, like visual perception and 3D movement, have been neglected. The main purpose of the experimentation in the simulator is to evaluate the performance of the dynamic hybrid position/force controller and the modifications made for it to be used on snake robots.

4.1.1 Motivation for the simulator choice

The customization of SnakeSIM towards the case of obstacle-aided locomotion is a crucial reason for why this particular simulator is chosen to conduct the experiments. It offers a setup with the desired frictionless environment, a simple snake robot and obstacles. A simpler version of a simulator with the same purpose was developed in MATLAB by the author [7]. However, this simulator lacked the integration of the physics describing the result of interaction between obstacles and the snake robot. The Gazebo physics engine in the SnakeSIM contributes to more realistic results in the experiments compared to what could be achieved with the MATLAB simulator.

Another MATLAB simulator tested is one developed by Transeth et al. [5]. This simulator is also designed for the simulation of OAL. Nevertheless, its

graphical user interface can not be compared to that of Gazebo. In addition, it is understood that the architecture of SnakeSIM is more modular, which allows for a smoother integration of additions to the program. This is again essentially due to the way ROS is designed. A more thorough explanation of ROS is provided in 4.2.

4.2 Simulator architecture

4.2.1 ROS

As mentioned above, the SnakeSIM simulator is based in and run from ROS, which is a set of open source software libraries and tools designed to build robot applications. The use of nodes ensures the modular architecture of ROS. A node is an independent part of the program that has a specific computational task. More specifically, it is an executable file from a ROS package. An example of this is the position controller node or the collisions node, detecting and reporting collisions. The nodes are in this project programmed using C++, but another supported alternative is Python. The modular architecture of ROS even allows for the nodes to be developed with different programming languages without any complications.

Even though the nodes can be run independently, they might need data from one another for useful computations. The nodes therefore communicate with each other through the use of topics. A topic is simply a set of variables that can be continuously updated by a node. Updating these variables is referred to as publishing to the topic. At the same time other nodes may read the values posted on this topic. This is done by subscribing to the topic. There

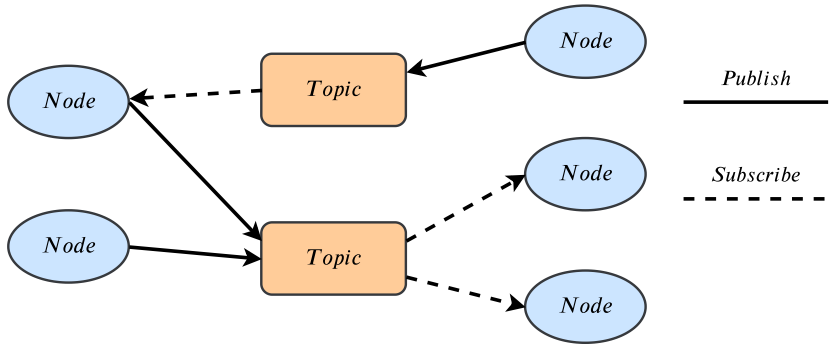


Figure 4.1: Message passing system in ROS

is no limit to how many nodes can publish and subscribe to a specific topic. An illustration of this message passing system is presented in Figure 4.1.

The nodes are able to find each other and communicate thanks to the ROS master. It registers all the running nodes and tracks topic publishing and subscriptions.

4.2.2 Gazebo

The ROS-based program is set up to launch the Gazebo 3D physics simulator. Gazebo is adapted to accurately simulate the snake robot in the desired environment with obstacles. All simulated objects, meaning the snake robot and the obstacles, have attributes like mass and velocity. This allows for realistic behaviour when interaction and collisions occur. The configuration of the simulated snake robot and obstacles is implemented according to the Universal Robotic Description Format (URDF) [36]. This is an XML-based file format used to describe elements like links, joints, sensors and actuators and how these elements connect to each other.

The added sensors are able to communicate properties like forces, torques, contact points, etc. These properties, as well as all physical variables, are reported back to ROS by Gazebo. At the same time, Gazebo subscribes to information from the ROS controller nodes sending joint motor torque commands to the snake robot. The physics engine is then able to use all this information to present a realistic resulting movement. It is also possible to move and influence the simulated objects directly through the graphical user interface (GUI) provided by Gazebo (see Figure 4.2).

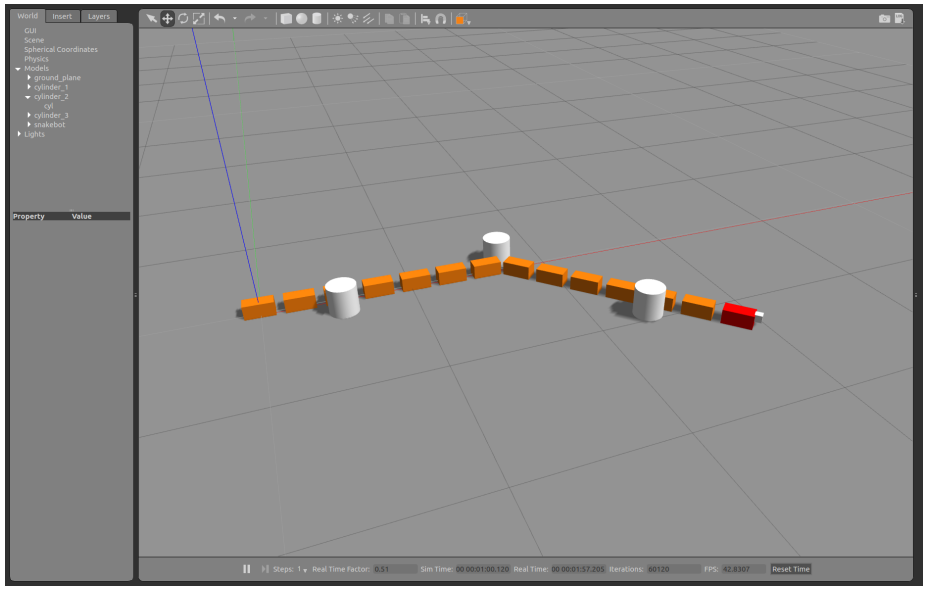


Figure 4.2: Gazebo graphical user interface

4.3 General simulation setup

In the past, the simulator has been used to mimic an existing snake robot prototype, namely the Mamba snake robot [37]. This snake robot has 14 identical links and 13 joints connecting these links. The visualization of the model can be seen in Figure 4.2. In order to make computations simpler and thus the real time factor of the simulation higher, the robot is scaled down to have 6 links and 5 joints for most of the experiments. As can be observed from the figure, the joints are not modeled as mechanical parts. They are therefore not visible, but placed mid-air between two neighboring links. Further model details can be found in Table 4.1. This table shows that the links of the snake robot model have three dimensional properties, meaning both width, length and height. All experiments are however still only conducted in the two dimensional ground plane.

	Value	**Unit**
Link mass	1	$[kg]$
Link length	0.2	$[m]$
Link height	0.1	$[m]$
Link width	0.1	$[m]$
Joint offset	0.3	$[m]$

Table 4.1: Simulated snake robot model properties

When it comes to the configuration of the obstacles, it is based on the *obstacle triplet model* [33]. The three obstacles are modeled as identical immovable cylinders in the simulator. Still, any contact with the snake robot is frictionless

point contact. The obstacles are positioned at the middle of the links on either the left or right side. Further details about the obstacle model can be found in Table 4.2.

	Value	**Unit**
Number of obstacles	3	
Mass	1	$[kg]$
Radius	0.1	$[m]$
Height	0.2	$[m]$

Table 4.2: Simulated obstacle model properties

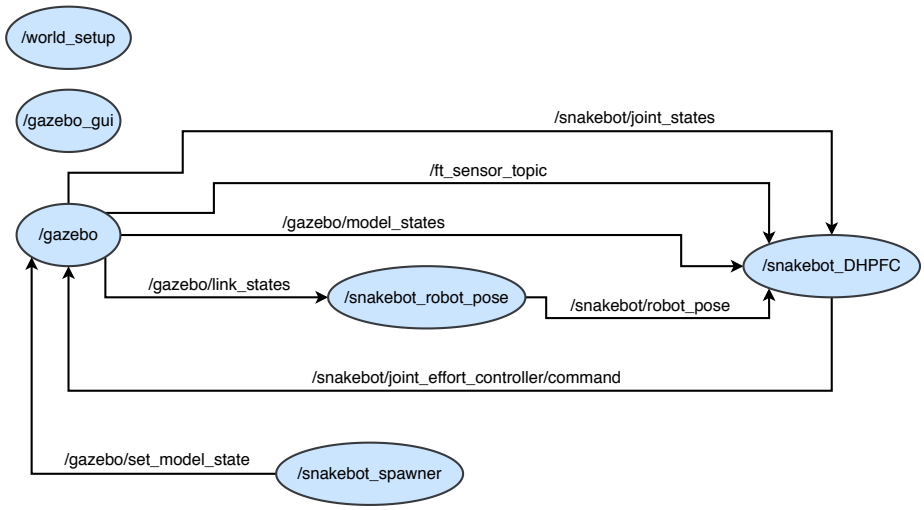


Figure 4.3: Overview of nodes and topics used in the project experiments

Not all available nodes have been employed for the experiments conducted in this project. An overview of the relevant nodes and the information flow

between them is presented in Figure 4.3. The nodes are depicted as blue ovals and the arrows describe the messages passed between them through topics.

Furthermore, a list of the most useful topics for high level control are listed below.

- `"/snakebot/pushpoints"`
  Information about the detected obstacle pushpoints. This includes contact normals and tangents, as well as position of contact point and information about which links are in contact with an obstacle.
- `"/snakebot/robot_pose"`
  Global position of all snake robot links.
- `"/snakebot/joint_states"`
  Angles of all joints relative to their preceding link. This topic also provides the corresponding velocity of the joints.
- `"/snakebot/joint_01_effort_controller/command"`
  Joint torque command to the specific joint actuators.
- `"/ft_sensor_topic_01"`
  Wrench in specific joint measured by the force/torque sensor in Gazebo.

Chapter 5

Simulations

This chapter presents four simulation experiments that aim at visualizing some of the theory presented in Chapter 3. The experiments cover the mapping onto the essential and allowable position space described in 3.5.4, the use of various closed kinematic chain formulations described in 3.5.3, the computation of control commands for active joints only, described in 3.2, and lastly the simultaneous control of position and force. All these experiments investigate modifications of the dynamic HPFC method presented in 3.1.2.

In addition to these control experiments, an experiment for validating the physics of the simulator is initially provided. Furthermore, the simulator setup and related limitations are presented to give a general overview of the experiment framework.

5.1 Simulator configuration for experiments

All experiments are executed in a sterile simulation environment only containing a frictionless ground, the snake robot and three obstacles. The simulator validation test is performed with a snake robot consisting of 14 links. The snake robot is however shrunk to six links for the remaining experiments. This is because the dynamic HPFC method relies on the computation of the dynamical model of the snake robot. This model is computed using the MATLAB Symbolic Math Toolbox [38] and MATLAB scripts developed in the previous project of the author [7].

Using the dynamical model of the snake robot is computationally expensive because the matrices for the equations of motion of the snake robot get very large for a large number of links. Furthermore, the MATLAB program is unable to calculate the dynamics of a snake robot with a large number of links within a tolerable time frame. Consequently, the control experiments are performed with a snake robot consisting of six links and five joints.

From the presented theory in earlier chapters it is known that the performance of HOAL in snake robots and generally the dynamic HPFC of snake robots is more ideal for snake robots with a large number of links. The fact that the snake robot consists of six links in the control experiments is therefore a limitation to the experiments. Another limitation is the placement of the obstacles. The computed mathematical model used for the control is calculated for a given obstacle-snake robot configuration. That is, the obstacles are in continuous contact with links 2, 3 and 5 on the right, left and right side respectively.

The initial configuration of the snake robot is the same for all experiments,

which is a stretched out position along the x-axis, starting in the origin. The joint angles do however deviate slightly from zero as a result of the physics simulator pushing the snake robot away from the static obstacles. This byproduct is probably also contributing to the encountered unsteady force sensor data from the snake robot. The initial configuration of the six link snake robot and the obstacles is shown in Figure 5.1. The green and blue lines in this figure are indications of axes and viewing angle in the simulator and can be ignored both here and in following simulator captures.

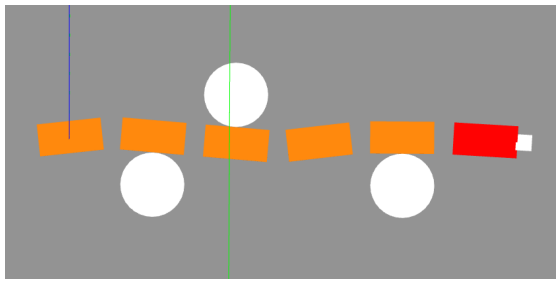


Figure 5.1: Initial snake robot configuration for dynamic HPFC experiments

The control experiments are all based on the dynamic HPFC from 3.1.2. The modifications and control structure for the different experiments are presented in the respective sections. All control goals are chosen as simple values to focus the results on the deployed methods. These desired values describe the force against the obstacles and the angle of links in contact with the obstacles. Only a subset of the variables are chosen to be controlled to their desired value for the different experiments.

5.2 Simulator validity test

The purpose of this experiment is to test the performance and validity of the physics engine in the simulator. In particular, the interaction forces arising from contact between the snake and obstacles are studied.

The idea is to control the snake robot to a completely stretched out stiff configuration while obstacles are symmetrically placed in contact around its center. At the same time, the snake robot should apply a constant motor torque to the center link. The obstacles are placed in a manner that prevents the joints from getting displaced. In order to still stay stretched out, all other joints will have to apply a torque calculated by the joint PID controller.

The simulator configuration for this experiment is summarized in Table 5.1. The link and obstacle specific configurations can be found in 4.3.

	Value	**Unit**
Number of obstacles	3	
Number of links	14	
Initial joint angles	$\mathbf{0}_{13\times1}$	$[rad]$
τ_7	-2	$[Nm]$
$[K_p, K_i, K_d]$	$[10, 3, 0.3]$	

Table 5.1: Simulation configuration for simulator validation test

The placement of the obstacles and snake robot is illustrated in Figure 5.2. As can be observed from the figure, the two outermost obstacles will establish a counter force to the resulting force from the middle joint motor torque. The resulting joint torques from this scenario are expected to comply with the well

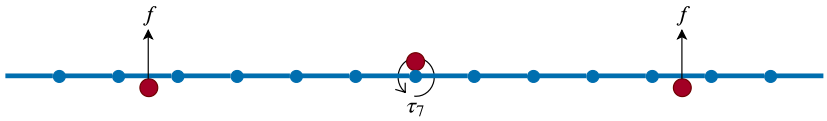


Figure 5.2: Illustration of validation test

known relationship

$$\tau = r \times f, \tag{5.1}$$

where r is the distance from the force origin to the joint.

Since all joints are separated with equal distances and the snake robot is completely stretched out, it is expected that the torques follow a linear relationship with respect to their placement from the center. In addition, since the obstacles are symmetrically placed around the center of the robot, it is expected that the torque values are symmetrical around the center joint as well.

The resulting joint torques are plotted in Figure 5.3. The x-axis of this plot describes the joint number from the center joint. In order to retrieve the joint torques from the static state part of the simulation, the data was recorded after the snake robot and controller had settled into a steady state.

From the figure it is obvious that the joint torques have a close to symmetrical and linear relationship. The fifth and sixth joints from the center are exceptions here. That is as expected since they are not between the outer and middle obstacles. A reason for why the graph is not completely linear or symmetrical is probably that it depends on the exact positioning of the obstacles and robot, and this might change slightly after the motor torques are applied. In addition, it depends to a large degree on the controllers effort to "stretch out" the snake robot. From studying Figure 5.4 it is possible to see that the snake robot is ever so slightly bent. Regardless, this experiment is considered to have

validated the interaction force computations of the simulator successfully.

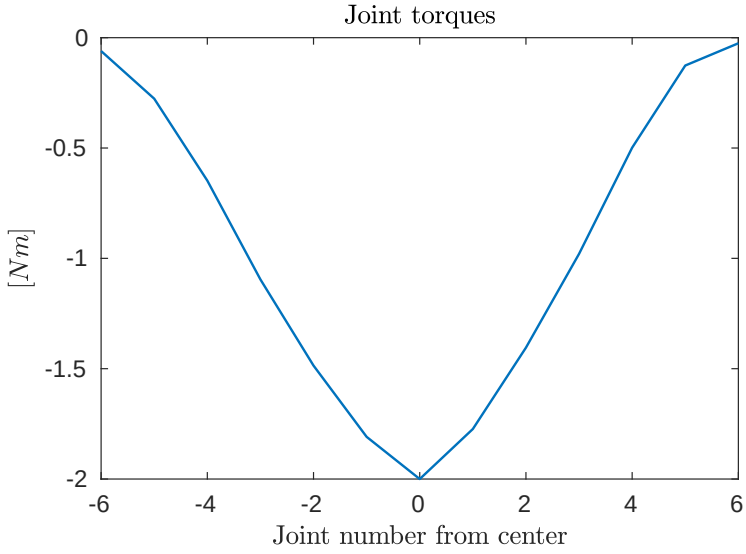


Figure 5.3: Snake robot joint torques from simulator validation test

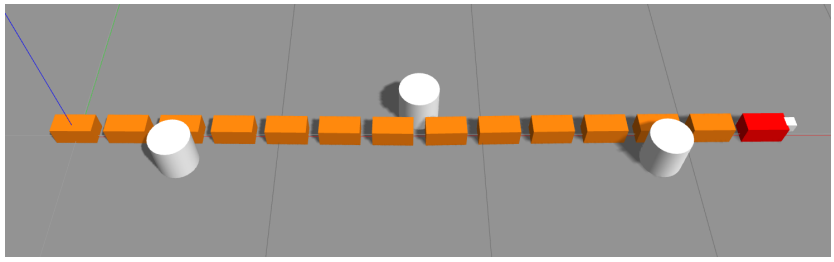


Figure 5.4: Screenshot from validation test Gazebo simulation

5.3 Essential position control

This experiment covers reference following for the angle of a link in contact with an obstacle. The motivation is to show the impact of the mapping to the joined essential and allowable position space. More specifically, the use of the filter ${}_j\mathbf{F}_p$ from (3.13) is studied. This method is also described in more detail in 3.5.4.

Link number 5, which is in contact with the foremost obstacle, is controlled to a constant angle $\theta_{t,d,3}$ measured with respect to the base frame. The controlled variable is referred to as $\theta_{t,3}$, and is the only essential variable the filter takes into account in this example. A simulation without the filter is also included to show the importance of defining which variables are essential for a given task when the total number of task variables is greater than the number of actuated joints.

	Value	**Unit**
Number of obstacles	3	
Number of links	6	
$\theta_{t,3,d}$	0.3	$[rad]$
$[K_p, K_i]$	$[0.05, 0.005]$	
Essential variables	$[\theta_{t,3}]$	$[rad]$

Table 5.2: Simulation configuration for position control experiment

The setup for this experiment is presented in Table 5.2. It should be noted that the force control is left out. This can also be seen in the control diagram of the experiment presented in Figure 5.5. The position filter is included in the

diagram, although it is only active for the first simulation of the experiment.

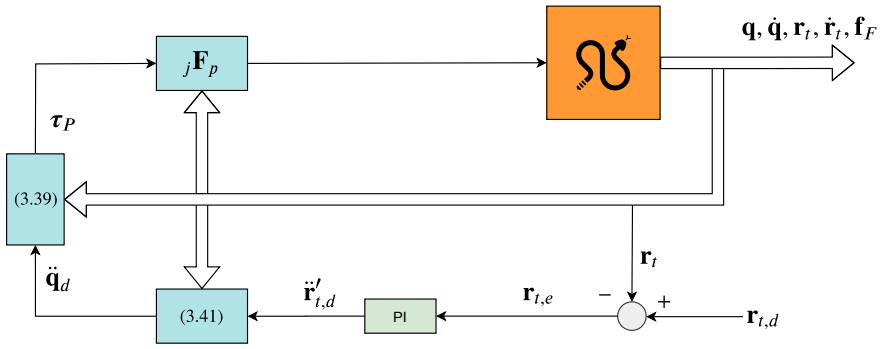


Figure 5.5: Control diagram for essential position control

The global contact link angle, joint angles and joint torques from the experiments are presented in Figures 5.6 and 5.7. As one could imagine, the latter figure shows the unfiltered control case. The contact link is in this case quite far from reaching its reference and the control does not seem to be very purposeful. The corresponding joint angle values show that the snake simply turns its first actuated joint, resulting in the tail of the snake robot spinning around before getting stuck. The behavior of the snake robot in the filtered example makes more sense, as it simply bends the joint preceding to the link and compensates by bending the following joint in the opposite direction.

The configuration of the snake robot at the end of the filtered and unfiltered simulations is better understood by looking at Figures 5.8 and 5.9 respectively. From the last figure it is also evident that the robot has lost contact with some of the obstacles, which means that the mathematical model the controller is based on no longer is valid. This is likely to have contributed to the strange control sequence.

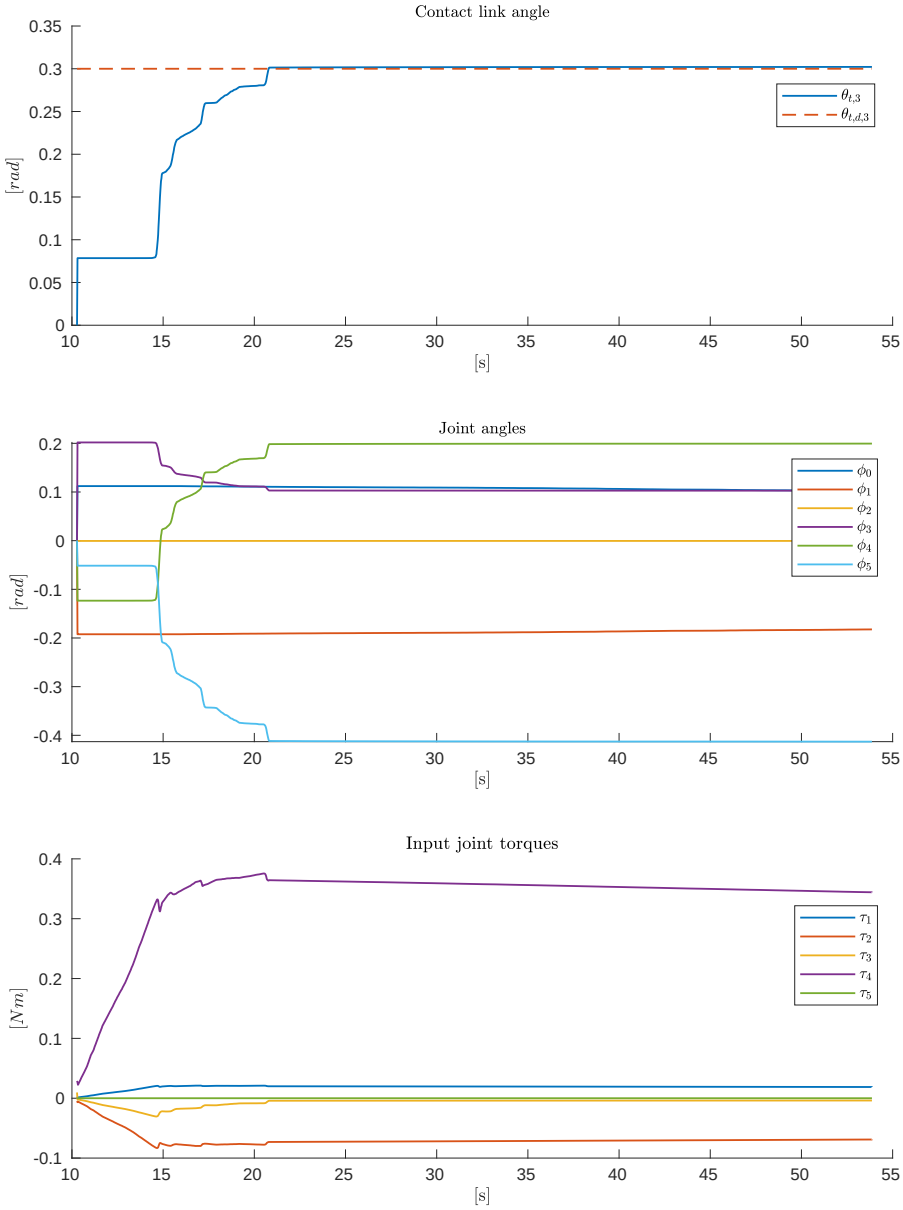


Figure 5.6: Results from filtered position control

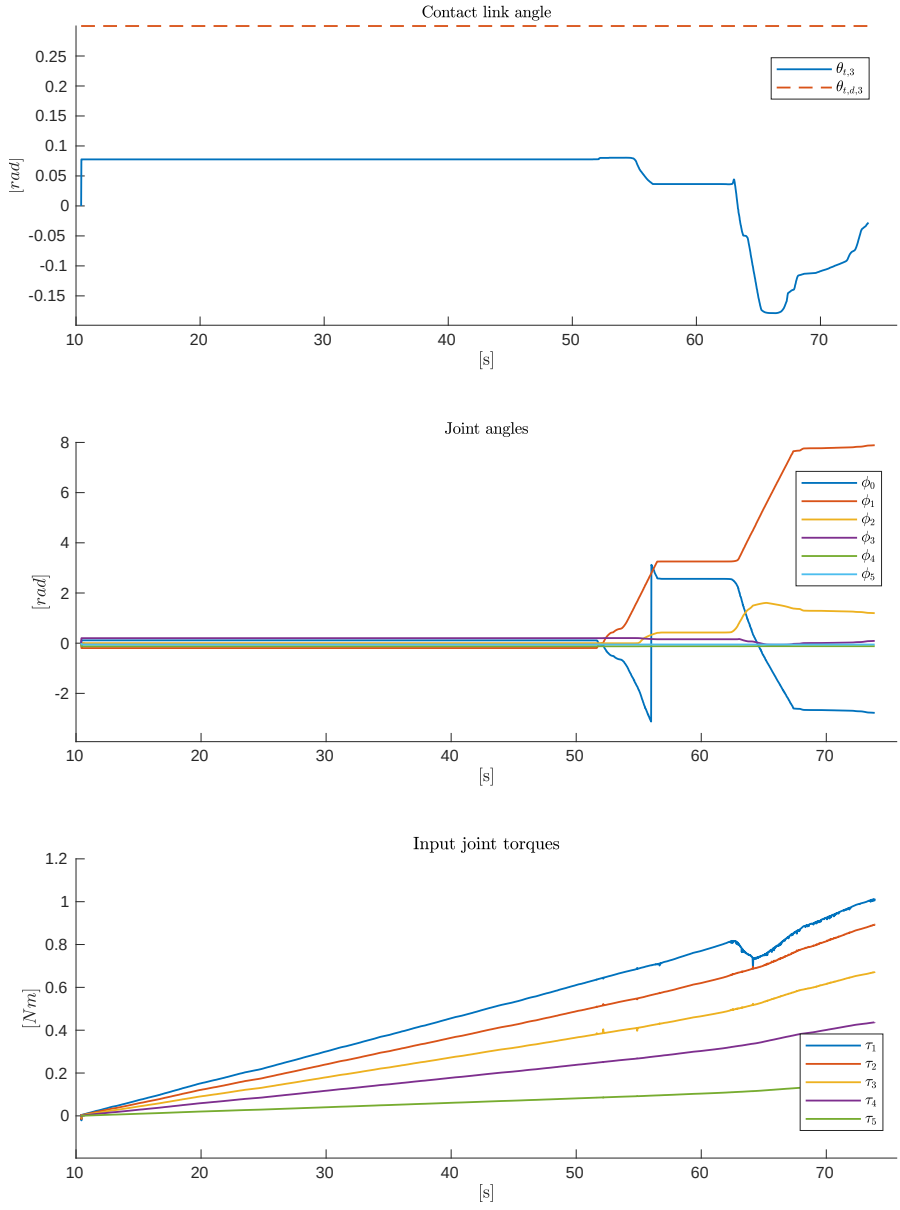


Figure 5.7: Results from unfiltered position control

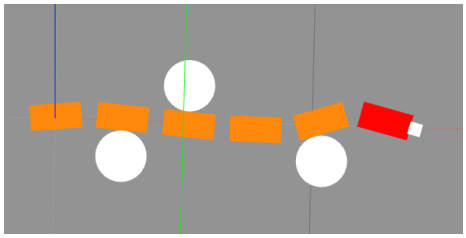


Figure 5.8: Snake robot after filtered position control

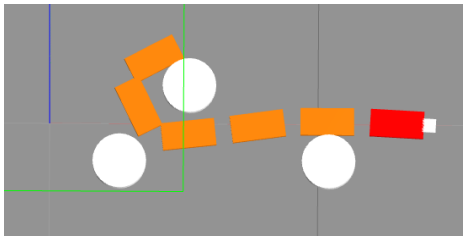


Figure 5.9: Snake robot after unfiltered position control

The explanation of the presented results is that the unfiltered case tries to control all possible position variables in **r**, meaning all contact link angles and the translational position along every contact point. This sums up to 6 variables. However, the robot only has 5 actuators. Furthermore, the last actuator can not be taken into account since it is located after the last contact point. This results in a total of 4 actuators that may be utilized for the control. Needless to say, trying to control 6 variables is infeasible for this snake robot. Increasing the number of joints and links would give it a much better basis for achieving the task.

5.4 Force control with various CKC formulations

The purpose of this experiment is illustrating the differences between the use of the two closed kinematic chain (CKC) formulations described in 3.5.3. The first simulation is run with minimal CKCs, and the second one is run with the regular CKCs defined with respect to the base frame.

	Value	**Unit**
Number of obstacles	3	
Number of links	6	
$f_{F,d}$	2	$[N]$
$[K_p, K_i]$	$[0.5, 0.003]$	

Table 5.3: Simulation configuration for force control experiment

The snake robot is in contact with three obstacles, whereas the force against the second and third obstacle is controlled. The desired force magnitude $f_{F,d}$ is

2 N for both contacts. The simulation configuration is summarized in Table 5.3 and is common for both simulations. The control structure for this experiment is given in Figure 5.10.

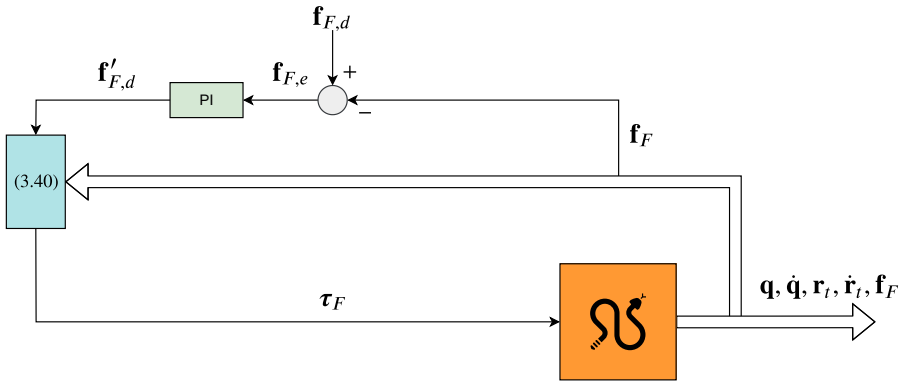


Figure 5.10: Control diagram for force control

It should be mentioned that very thorough tuning of control parameters could improve the results of both simulations, although a great deal of tuning already has been conducted. Furthermore, the control is quite twitchy as a result of the unsteady force sensor feedback. The experiment in 5.6 is conducted with low pass filtering of these sensor signals, and proves that the control consequently gets smoother as well.

5.4.1 Regular CKCs

For the first simulation, the regular CKC formulations are used. This means that all contact points are described with respect to the base frame of the robot, as illustrated in Figure 3.6 in 3.5.3. It can be seen that the two CKCs for the second and third obstacle are overlapping. As a result, the first two joint motors will be used to control both contact points. The motors for joints 3 and 4 will however still be reserved to the control of the third contact point. From the input torque plot in Figure 5.11 it can be observed that all joint motors except for the last one are actuated. The last one is left out as it is positioned after both controlled contact points.

Figure 5.11 also presents the resulting contact forces. As expected, the contact force for the third contact point is much better controlled than for the second contact point. This is because it has more actuators available, again making it more robust. In addition, it is known that the actuators used for the second contact point are shared for both controls. Consequently, the input torques τ_3 and τ_4 used only for the third contact are much more stable than the shared input torques τ_1 and τ_2.

Another reason for why $f_{F,2}$ never reaches its desired value can be that the joint torques related to this control reach the saturation limit, which has an absolute value of 2 in this experiment. Unfortunately, it was found that larger motor torques easily led to the contact being lost at some moments. This is highly undesired as the mathematical model assumes contact at all times.

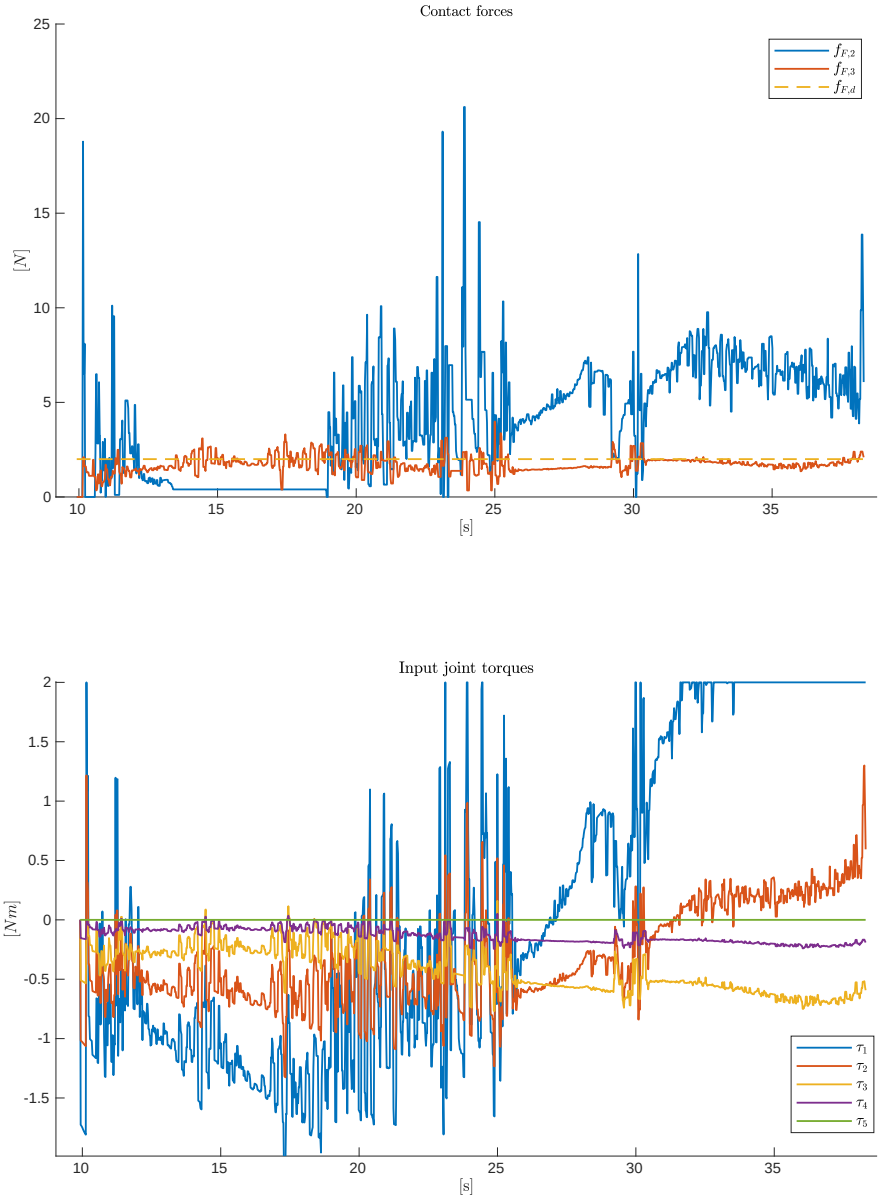


Figure 5.11: Results from experiment with regular CKCs

5.4.2 Minimal CKCs

For the second simulation, the minimal CKC formulations are used, meaning that the CKC for the second obstacle is defined from the first to the second obstacle point, and the CKC for the third obstacle is defined from the second to the third obstacle. This formulation is easier understood from Figure 3.7 in 3.5.3. It can be observed that the CKC belonging to the third obstacle contains two joints, whereas the second one only contains one joint.

Figure 5.12 shows the resulting contact forces and joint torques from the simulation. It can be observed that both forces are close to the desired value. Nonetheless, the force $f_{F,3}$ against the third obstacle is considerably more stable than the force $f_{F,2}$ against the second obstacle. It is not implausible that this is a result of the third contact point having one more joint available for control. This increases the robustness of the control. In addition, it is known that the joint connecting link 3 and 4 will influence the third link, which is in contact with the second obstacle. This can further be seen as a disturbance on the control of the second contact force.

From the control torques in Figure 5.12, it is evident that only the motors on joints 2, 3 and 4 are used. This is logical, as it is exactly what the corresponding CKCs allow. Since the controllability increases with the current minimal CKC formulation, the control is also significantly smoother than in the previous simulation.

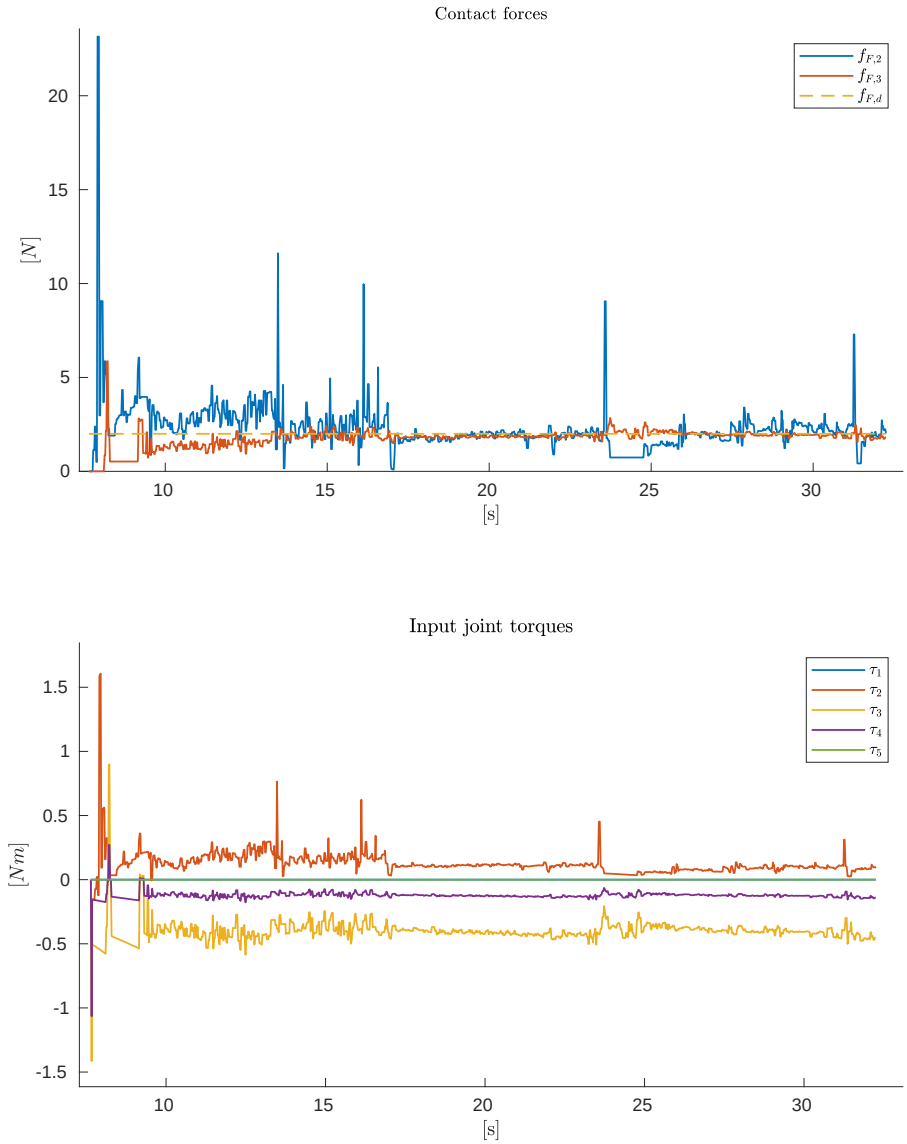


Figure 5.12: Results from experiment with minimal CKCs

5.5 Position control with active joint torque focus

As seen in 3.1.2, the equations of motion are used to find the desired torques satisfying the desired joint accelerations. However, the equations of motion do not consider that some of the joints are passive and thus joint torques will be assigned to all joints, both passive and active. This is of course an issue since the computed solution relies on the realization of these torques.

	Value	**Unit**
Number of obstacles	3	
Number of links	6	
$\theta_{2,d}$	-0.2	$[rad]$
$\theta_{3,d}$	0.2	$[rad]$
First sim.:$[K_p, K_i]$	$[3, 0.01]$	
Second sim.:$[K_p, K_i]$	$[1, 0.001]$	

Table 5.4: Simulation configuration for double position control experiment

This experiment aims at illustrating the difference in just ignoring the passive joint torque commands and in mapping all joint torques over to the active joints. Because the focus is on the position torques $\boldsymbol{\tau}_P$, the link angles by the second and third obstacles are controlled. They are first controlled by using the equations of motion as in (3.39). Here the torques for the passive joints are simply neglected. The second simulation shows control where all torques are mapped to the active joints, as explained in 3.2. For this experiment it is simply chosen that the desired values of the active joints should be followed and that the passive joints could take resulting arbitrary values. This is probably not the

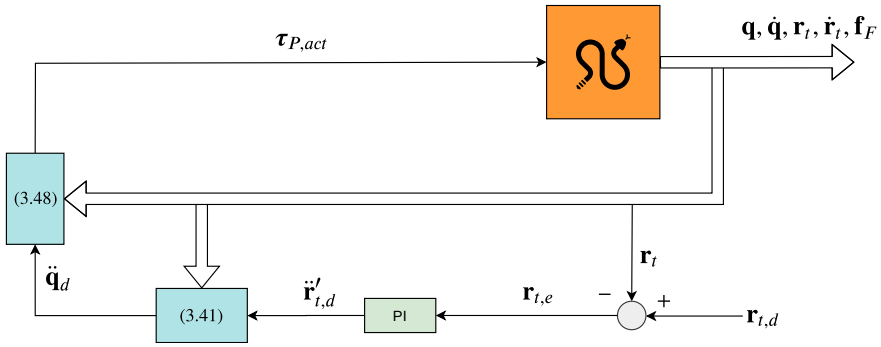


Figure 5.13: Control diagram for position control with active joint torque focus

optimal choice of variables to control, but a method for explicitly determining this has not yet been developed. The matter is discussed further in 6.1.3.

The minimal CKC formulation is used for both simulations, which implies that the desired angles are defined according to the previous contact point. Further simulation configurations are presented in Table 5.4. The control structure for the experiment is given in Figure 5.13. The equation (3.48) in the diagram is substituted with (3.39) for the first simulation.

5.5.1 Control torques computed for both passive and active joints

For this simulation, the control torques from (3.39) belonging to the active variables were commanded to the snake robot. The rest of the motor torque commands were simply ignored. The resulting torques and contact link angles can be seen in Figure 5.14. The contact link angle $\theta_{t,3}$ settles at a value close to its desired value. However, $\theta_{t,2}$ is very far from reaching its desired value and

the control is overall considered unsuccessful.

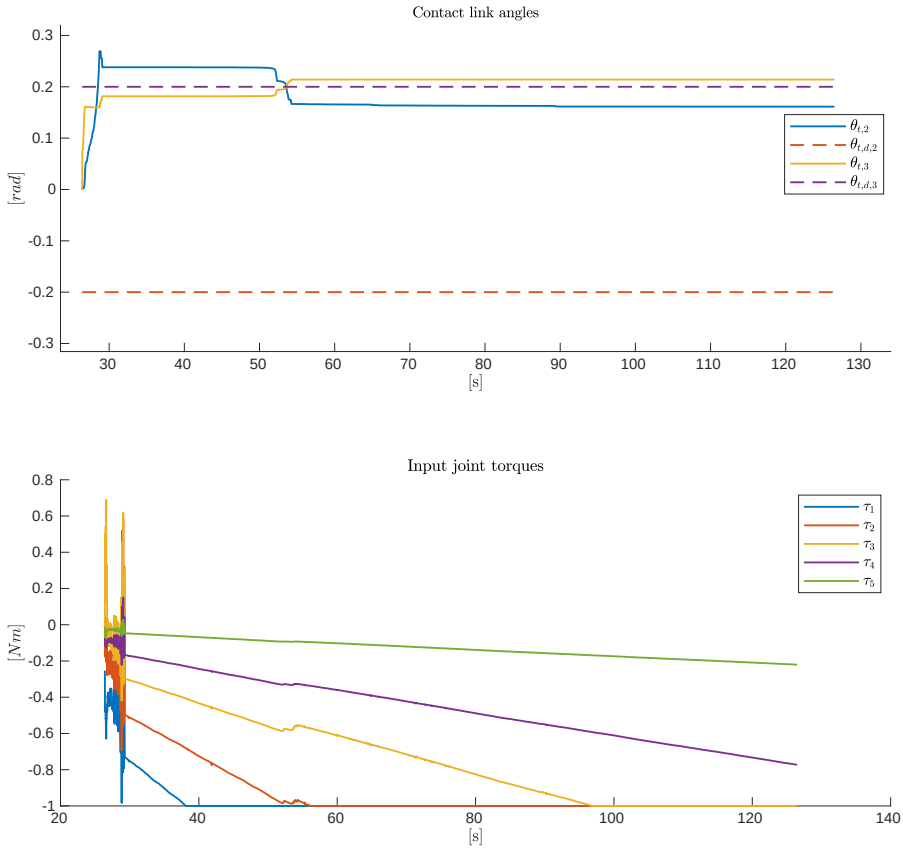


Figure 5.14: Results from position control experiment with torque calculation for all joints

From Figure 5.14 it can also be observed that the joint torques reach a saturation limit, which is probably the reason for why very little change is

noticed in the joint angles. The saturation is in place to keep the snake robot from making large sudden movements that would move it outside of the scope of the model. A higher saturation limit is also tested without further success. Another reason for the static behavior of the angles is the configuration of the snake robot and obstacles, presented in 5.1. When all applied joint torques are negative the snake robot will try to curve up towards the two right side obstacles. By doing so it is mainly just pushing against the obstacles, much like the experiment in 5.1. This will logically lead to a jam rather than any movement.

5.5.2 Control torques computed for only active joints

This time all control torques are calculated to be commanded to the active joints, which means that no commands are ignored. The desired passive joint acceleration values are on the other hand ignored. Therefore, this is not an optimal approach either and required a lot of tuning to get it right. The results are presented in Figure 5.15. $\theta_{t,2}$ still deviates slightly from its desired value $\theta_{t,d,2}$, but the performance is considered much better than in the previous simulation.

It should be noted that the snake robot is moving quite slowly in both of the simulations. This means that $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, the part of the dynamics dependent on the joint velocities, is playing a very inessential role here. To investigate this method more thoroughly, further experiments should be carried out with a larger number of snake robot joints, more rapid movements and last but not least, a more intelligent and reasoned choice of joints that should be precisely controlled.

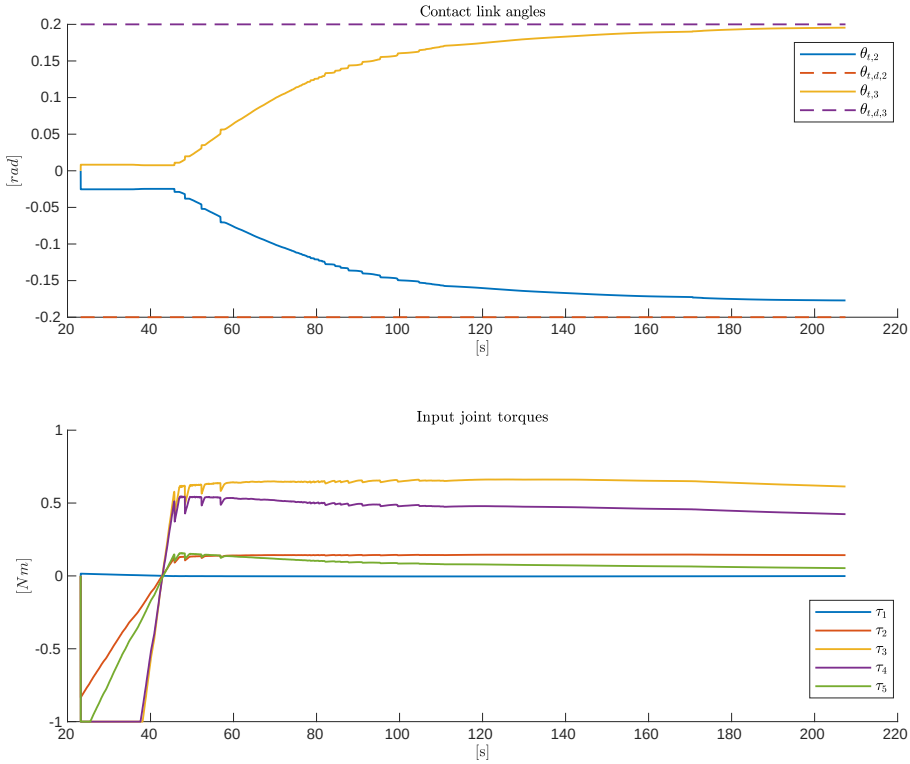


Figure 5.15: Results from position control experiment with torque calculation for only active joints

5.6 Simultaneous position/force control

This last example combines the explored methods from the previous experiments 5.4-5.5. That means that the minimal CKC formulation explained in 3.5.3 is applied, and that the joint torques are computed for only the active joints as explained in 3.2. The rest of the control method is according to the dynamic HPFC control scheme presented in 3.1.2.

	Value	**Unit**
Number of obstacles	3	
Number of links	6	
$f_{F,d,3}$	1	$[N]$
$\theta_{t,d,3}$	0.1	$[rad]$
Force:$[K_p, K_i]$	$[1, 0.005]$	
Position:$[K_p, K_i]$	$[3, 0.005]$	

Table 5.5: Simulation configuration for simultaneous position and force control experiment

In this experiment, both position and force is controlled for the third contact point. More specifically, the angle of the link in contact with the third obstacle and the force this link applies to the obstacle are controlled. The exact desired values, as well as general simulation configurations for the experiment, are given in Table 5.5. The values were chosen with the intuition of what would be achievable for the snake robot from its initial configuration. The control structure for the simulation is given in Figure 5.16.

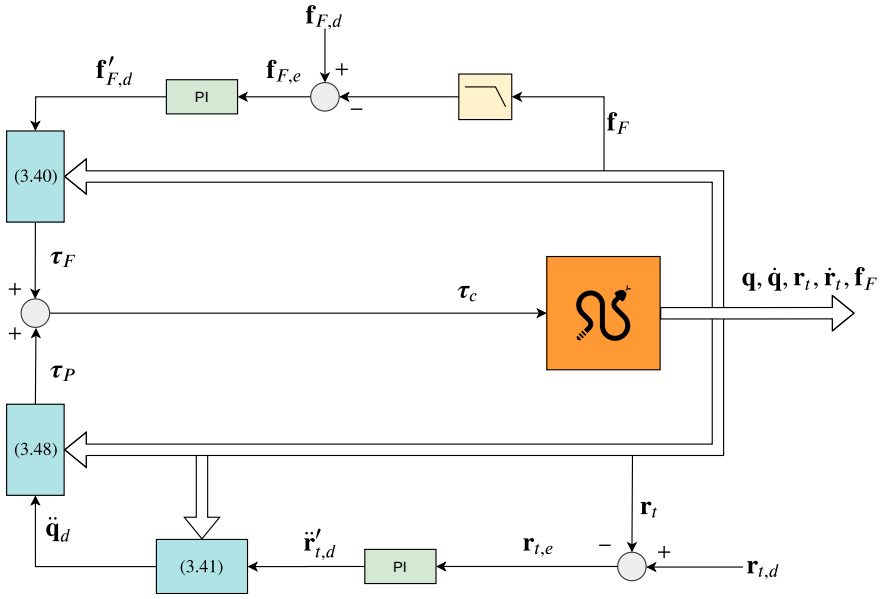


Figure 5.16: Control diagram for dynamic HPFC

The resulting contact link angle, contact force and joint torques are presented in Figure 5.17. In this experiment, the torque values controlling the force are much smoother than the force signal, as opposed to the experiment in 5.4. This is because the force sensor signal is filtered before it is sent to the controller. The force signal shown in the figure is unfiltered.

From Figure 5.17 it can also be seen that both the desired angle and force is reached. However, the angle takes longer to reach its reference and does have a slightly irregular trajectory. Both this and the very sensitive force signals can be a result of the nature of the simulator used.

From the input torque plot in Figure 5.17 it can be observed that the commanded torques τ_1, τ_2 and τ_3 are used only for the position control since they are smoother, whereas τ_4 and τ_5 are used for both position and force control. The idea of Yoshikawa [10] is that this can still yield successful HPFC given that the dynamical model is correct. The position and force loops are included to compensate for possible modeling errors.

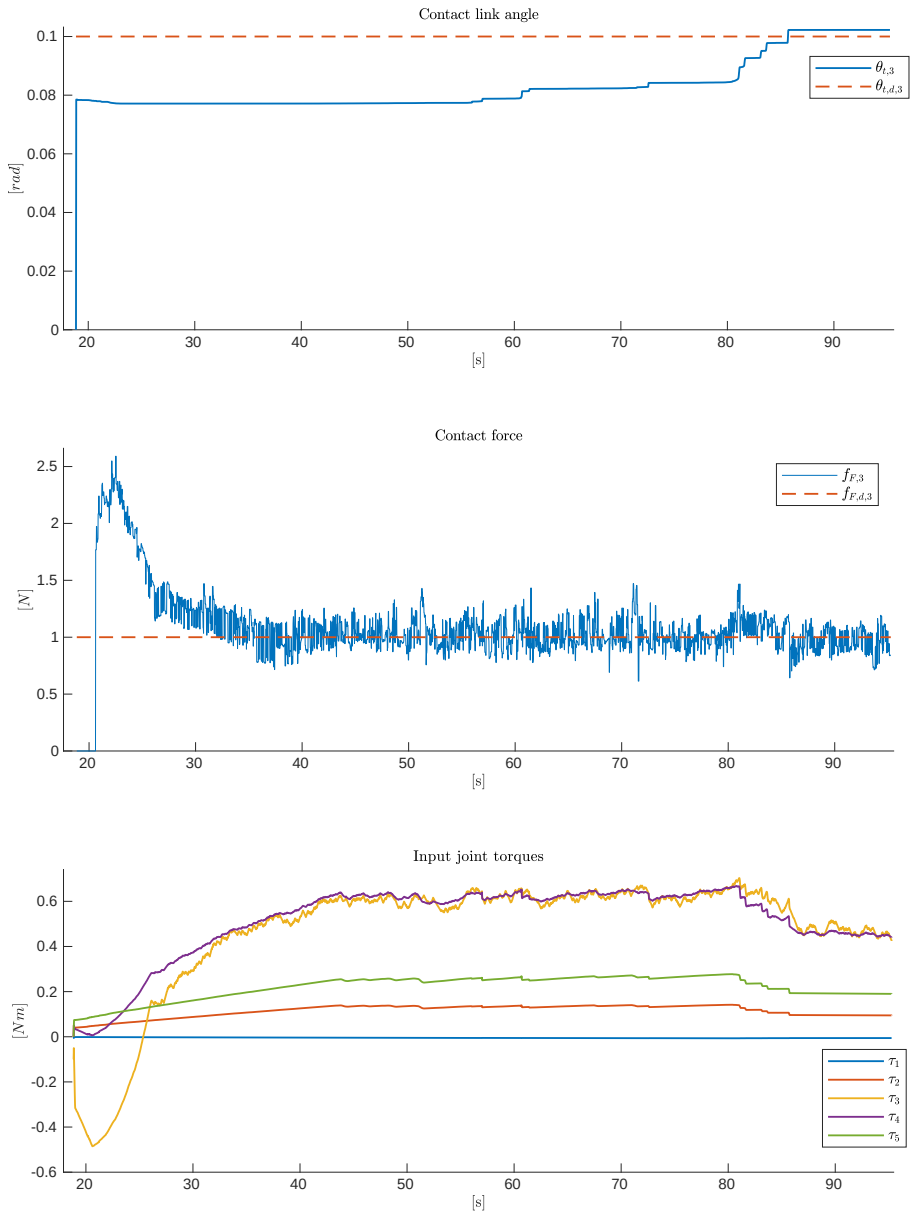


Figure 5.17: Results from simultaneous force and position control

Chapter 6

Discussion

This chapter describes the effects of the different modifications and adaptions made to the dynamic HPFC method. The discussion is based on the performed experiments in Chapter 5 and the mathematical analysis carried out in Chapter 3. It further evaluates the chosen simulator, SnakeSIM, and the implementation of the dynamic HPFC method in this simulator. Lastly, the application and requirements of the dynamic HPFC method for HOAL are assessed. Some improvements to the mentioned topics are proposed throughout the chapter.

6.1 The snake robot dynamic HPFC method

This section discusses the modifications made to the dynamic HPFC method in order to adapt it to snake robots.

6.1.1 Differences with traditional HPFC

As the name implies the dynamic HPFC method of Yoshikawa [10] described in 3.1.2 depends on the dynamical model of the robot. This is the main difference from the plain HPFC method of West and Asada [8] described in 3.1.1, and makes the calculations significantly more complex. It is however believed that it pays off in the resulting ability to predict the dynamical behaviour of the snake robot.

In this project, the dynamics are calculated using the Euler Lagrange method. It was discovered that the computations get quite extensive for snake robots with a large amount of links. An alternative method is provided by Liljebäck et al. [2], which is significantly easier to apply to snake robots with many links. The disadvantage with this method is however that it is based on the center of mass of the snake robot, which is an irrelevant parameter for the HOAL problem. An option is adapting this method to depend on the position of the tail instead, which is already part of the generalized coordinates in the presented snake robot model.

The assumption of no friction made in this project surely simplifies the snake robot dynamical calculations. On the other hand, it is known that friction contributes to dissipation of energy, which again naturally contributes to stability of the system. At the same time it can lead to control deviations. Regardless of the merits and disadvantages, friction plays a significant role in physical situations and should therefore later be included in the model.

The traditional HPFC method is able to analyze whether or not a task is achievable by mapping onto the joined space of allowable and essential directions of the task. These essential directions have to be defined by the user.

For dynamic HPFC, the allowable motion and force directions have to be predefined as well, but the user or planner algorithm additionally has to consider task combinations that are in fact achievable. For instance, the snake robot might be in a position to push against and shape itself along five obstacles, but depending on the composition of the snake robot and the distribution of these obstacles, achieving all control goals simultaneously might not be feasible.

A possible solution is evaluating the mapping filters (3.13) and (3.14) of West and Asada [8] and the property that they have zero rank if the task is outside the combined allowable and essential spaces. Ideally, the information gained from these filters should be exploited by the HOAL planner algorithm, leaving the controller to simply realise any task it is requested to.

6.1.2 Mathematical formulations

It should be mentioned that since it is the first time the dynamic HPFC method has been adapted to snake robots, the chosen formulations can still be challenged. The formulation of the contact points are based on the idea of which variables the HOAL algorithm will desire to control. That is the global orientation of the contact links and the directions along and against the obstacles. When the minimal CKC formulation was implemented and used to compute the Jacobians, these variables were no longer all defined with respect to the base frame, but rather the frame of the preceding contact point. The intuition behind what the desired variables should be was then somewhat lost. Still, this can easily be solved by defining all desired values with respect to the base frame and use homogeneous transformation matrices to input the right formats to the controller.

6.1.3 Passive joints

An issue that was encountered during the testing is that the dynamic HPFC method does not consider that some of the joints are passive and thus unactuated. Consequently, it computes motor commands for these joints as well as for the active joints. These commands cannot be realised, and merely ignoring them leads to unwanted behaviours, as was seen in the experiment in 5.5. This observation motivated the suggested solution described in 3.2, where a subset of the joint variables ($\boldsymbol{\psi}$) are picked for reference following and the generalized control forces are calculated only for the active joints. This was tested and presented in 5.5, and led to a considerable improvement in the results.

It is however not believed that this is the most optimal solution to the problem, because it requires continuous determination of the variable subset $\boldsymbol{\psi}$ and rearrangement of the generalized coordinates and dynamics matrices. There has so far not been developed any explicit method for determining the subset of the joint variables that are the most significant for achieving a given task, and it is an essential matter that should be investigated further. A lot of trial and error, both with the variable subset and tuning of the internal controllers, was conducted before satisfying results were attained in the experiment in 5.5.

Furthermore, the arbitrariness term $(\mathbf{I} - \mathbf{J}_t^+ \mathbf{J}_t)\mathbf{k}$ in (3.41) for the calculation of the desired joint accelerations $\ddot{\mathbf{q}}_d$ has not been explored extensively and thus not been included in the experiments. It is also more significant for snake robots with a large number of links where the arbitrariness of the motion is higher. However, one idea is combining the choice of $\mathbf{k}$ with the choice of the subset $\boldsymbol{\psi}$ of the joint variables mentioned above. If the term $(\mathbf{I} - \mathbf{J}_t^+ \mathbf{J}_t)\mathbf{k}$ can influence which joint variables are assigned the most vital values for reaching

a goal, then $\boldsymbol{\psi}$ could be altered less frequently. This is simply a hypotheses, and other ways of exploiting the influence of $\mathbf{k}$ should be studied as well. It should also be noted that since the vector $\mathbf{k}$ will be projected onto the nullspace of the robot, there are a lot of variations of $\mathbf{k}$ which will yield the same resulting projected vector. This decreases the span of variations that have to be investigated.

6.1.4 Closed kinematic chains

The analysis of the closed kinematic chains (CKCs) in 3.5.3 concludes that a higher number of joints leads to a higher level of controllability. Because the variables desired to control are related to the contact points, the arrangement of the obstacles in touch with the snake robot have an impact on the controllability as well. That is, if for instance all obstacles are gathered at the back part of the snake robot, it will not be able to exploit all of its joints to control the contact point variables. In other words, it is desired to have a snake robot with a large number of joints configured so that there are sufficiently many joints between every contact point to make control of the contacts close to independent of each other. A higher number of joints also allows for a higher number of solutions to every control problem and will thus increase the dexterity.

In case it is not important to control all contact points accurately, one should implement an algorithm that can find the largest CKC for every contact point desired to control that does not overlap with any other active CKCs. This way the utilization of the snake robot joints can be maximized and the controllability increased. On the other hand, it is sensible to always control all contact forces based on the fact that a mechanical snake robot can break or get damaged

under too much pressure.

6.1.5 Control structure

When the complexity of a controller increases, it is typically dependent on a greater amount of feedback parameters from the system to execute all of its calculations. Thus, it is dependent on the accuracy of a greater amount of parameters. The dynamic hybrid position/force controller for the simulated snake robot has access to accurate data of position, velocity and force. These parameters are used both for the dynamic HPFC and for the inner control loops considering the desired forces and positions. Even though the data was considered to be of good quality, considerable tuning had to be conducted to optimize the inner control loops and approach an optimal controller.

For this project, the velocities and displacements have been kept small in order to stay within the validity bounds of the mathematical model of the snake robot and its constraints, and the control parameters are tuned correspondingly. A more thorough tuning of control parameters should thus later be conducted for a faster, yet stable, response of the system.

The inner control loops, which consider the position and force errors, can be utilized to weigh the control of the different variables. I.e., it is possible to control some variables more strictly than others if they are recognized as more vital for achieving the propulsion goal. To figure out which variables are the most important, the necessary conditions for propulsion should be explicitly determined and analyzed.

Furthermore, PI control was chosen for the inner control loops since all motion was very slow either way. It is however suggested to implement PID or

PD controllers in the continuation of the research of this project. In addition, the desired contact point accelerations were calculated directly from the desired position variables, rather than the velocities. It is not believed to have had any negative consequences in the scope of this project, but should also be altered in future implementations.

Since singularities have not turned out to be prominent in the vicinity of the experiments in this project, singularity avoidance has not been included in the control. It should however be implemented when the scope of the testing is increased.

6.2 Simulation limitations

6.2.1 Snake robot model

The most important deficiency in the quality of the simulation experiments is that the number of links is quite low. The reason for this is explained in 6.1.1. In future experiments it is recommended to use a snake robot with a large number of links so that the controllability increases and artefacts or disturbances from control of adjacent contact points are minimal.

Another disadvantage is that the movements have to stay within the bounds of the model description, meaning contact with the links have to be maintained at all times. It was discovered that uncontrolled oscillations could occur when the snake robot violated this assumption. When the movement span increases it is also more likely that the snake robot ends up in or approaches a singular configuration, which leads to large joint velocities. However, the purpose of the experiments in this project is not achieving propulsion, but to test the dynamic

HPFC method on a snake robot.

Another resulting limitation to the test quality is that the dynamical model is less significant and influential for very slow movements where the velocities are close to zero. Therefore, the dynamical part of the implemented method has not been tested as extensively as it could have been in a different test and model environment.

6.2.2 Force sensor signal

The force sensor signal from the physics simulator is notably more unsteady than the other signals provided. A reason for this can be that calculating forces in collisions is very complex. The links in contact did not collide with the obstacles very frequently during simulations. However, it is believed that the unsteadiness of the signal is a result of the contact point on the contact link constantly changing. The displacements might not be big, but enough for the simulator to register a new contact point and thus recalculate the force with new parameters. It was observed that when the robot was lying completely still the force signals were much steadier. According to Guillaume [39], continuous contact is excluded because contact will only happen on the triangle edges of an object.

Despite the unsteady force signals, the controller is able to drive the contact forces to oscillate around the desired value. Lowpass filtering the force sensor signals was further observed to have a positive impact on the control, as can be seen in the experiment in 5.6.

An unfortunate trait of the physics simulator is that it has a tendency to push the snake robot ever so slightly away from the obstacles when it is lying

still. The movement is not significant, but the problem is that it results in the contact force signal and information about the contact point being lost.

6.2.3 Simulator user-friendliness

Using the ROS/Gazebo SnakeSIM simulator is not quite plug-n-play, and it is recommended to reserve enough time to understand the principles of the program structure and how it should be used. Both knowledge about the Ubuntu operating system and the use of the command line is required.

The previously made modules for this simulator are well developed and it intuitively follows which modules should be deployed. On the other hand, there is lacking documentation of the different modules and the simulator in general, which made familiarization with the program very time consuming.

6.3 Dynamic HPFC for HOAL

The experiment in 5.6 shows that simultaneous control of position and force against a single obstacle is achievable with the presented modifications to the dynamic HPFC method. This experiment is, however, not enough to determine whether or not dynamic HPFC is the best method to be used for HOAL. Further experiments with larger velocities, a higher number of snake robot links and simultaneous control of several contact points should be carried out to substantiate the idea. It can, however, be concluded that the results are an indication that the research is moving in the right direction.

Furthermore, it has been shown that the amount of links, and in particular the amount of links between contact points, is vital for the performance of the dynamic HPFC method and thus the HOAL scheme. The results of the

experiment in 5.4 suggest that at least two actuated joints are needed for accurate control of the force at one contact point. It is also achievable with one actuator, given that the snake robot is able to push against another obstacle to stay in place, but this is less optimal. An important factor that should be considered is that the control situations and goals were very simple in this project, and the minimum requirement might thus increase for more complex scenarios. Ideally, hyperredundant snake robots should be used, as described in 3.5.3.

In order to focus the control on variables that are in fact desired to be controlled by a given HOAL algorithm, it was suggested in 3.5.4 that the position and force filters from the traditional HPFC method can be combined with the dynamic HPFC method. This was proven to be successful in the experiment described in 5.3. However, several conditions and requirements have to be fulfilled for the proposed method to be valid, and it is therefore not believed that it is scalable to bigger experiments with higher velocities. As mentioned in 6.1.1, it should still be investigated if these filters can be exploited by any part of the HOAL algorithm.

Chapter 7

Conclusion

This project has studied the adaptation and application of dynamic HPFC on snake robots to allow for both natural and virtual constraints on position and force to be met during propulsion. The focus has been on controlling variables that are considered necessary for achieving HOAL. That includes the position and orientation of the snake robot alongside obstacles and the contact force between the snake robot and obstacles. A lot of the contributions to the research topic have been formulations and definitions for the snake robot dynamic HPFC problem. This chapter concludes the insights from the simulations of the dynamic HPFC method and the developed modifications to the method for it to be suitable for snake robots performing HOAL.

Surely, a great deal of further testing is required to support the dynamic HPFC method for snake robots. There is also room for improvements, both when it comes to the mathematical foundation and the test environment. These aspects are regarded in Chapter 8.

7.1 Insights from simulations

From the experiments it can be concluded that the modifications made to the dynamic HPFC method are not alone sufficient for controlling the several contact points of a snake robot independently. This is especially the case for snake robots with a moderate number of joints. However, the performance was considerably increased with the developed additions that assign certain actuated joints to certain contacts so that the control of the different contacts do not overlap.

The control managed to steer the position and force to follow constant reference values within small bounds of the initial configuration of the snake robot. The low number of snake robot joints limited the experiments and only very specific scenarios could be tested. It can be concluded that the number of snake robot links relative to the number of obstacles is very essential for controllability. More specifically, the number of links between every obstacle contact is important. This was investigated through the study of the different closed kinematic chains present in the snake robot and the inclusion of these in the control.

One solution for considering the presence of passive unactuated joints based on the dynamic coupling of the snake robot was presented and tested. From the experiment it is obvious that some adaption of the dynamical joint torque calculation is indeed necessary, although this method might not be the most ideal, as discussed in Chapter 6. The solution should be developed further and adapted more thoroughly to the snake robot case.

Chapter 8

Future work

This chapter proposes some ideas for future research within the field of HOAL, with special emphasis on the dynamic HPFC method. Possible improvements based on the challenges encountered in this project and the analysis carried out in Chapter 3 are suggested.

8.1 Dynamic HPFC method

As mentioned earlier, further testing should be carried out to evaluate the developed method for dynamic HPFC of snake robots. Since it has been understood that snake robots with a large number of links are ideal for the HOAL scheme, it is desired to conduct further testing with a much bigger snake robot. The tests should still be conducted in an enclosed simulator test environment so that different aspects can be analyzed under isolation of unknown outer disturbances and with accurate position and force data

available. The SnakeSIM simulator is recommended for further tests as well. However, a better method of calculating the dynamics of a snake robot with arbitrarily many links should be implemented.

Furthermore, with a higher number of links, and thus an increased dexterity level, a greater amount of solutions to the control problem will present themselves. It is therefore desired to analyze the arbitrariness of the behaviour of hyperredundant snake robots. Special emphasis should be put on addressing the arbitrariness term in the calculation of the desired joint accelerations.

A better solution for isolating the commanded control torques to only apply to actuated joints should also be investigated. By preference, the proposed solution can be adopted, but it is recommended to identify a more mathematical and automatic way of finding exactly which joints should be controlled precisely and which can take arbitrary values resulting from the controlled joints. A proposition is looking at the span of the snake robot propulsion space and the desired velocity directions along the given path. Seeing as a necessary condition for propulsion is that the velocity direction along the path lies within the propulsion space, the underlying criteria for this could be analyzed and used to study the influence and importance of the different joints, both active and passive.

8.2 HOAL

When the dynamic HPFC method for snake robots has been established, it can be combined with the suggested HOAL algorithm. There are of course several other parts that need to be studied and established as well. This includes the development of automatic path planner and path following algorithms.

Section 3.6.2 discusses some of the necessary criteria for propulsion that should be considered by these algorithms. An important topic that needs to be investigated further is the span of the different spaces, namely the shape, constraint and propulsion space of the snake robot. From this, criteria will follow for how the desired path should be designed with respect to the given snake robot and obstacles in its environment. Additional criteria that could benefit the snake robot locomotion is minimizing the energy consumption (discussed by Holden et al. [40], [19]) and the traversed distance. Methods like model predictive control (MPC) and reinforcement learning (RL) are suggested to be investigated in future work. Lastly, the inclusion of the position of the snake robot head to the dynamic HPFC task space vector $\mathbf{r}_t$ could be useful for the path following component.

8.3 Simulation platform

The research field of snake robot OAL/HOAL is growing, and it is therefore believed that a common robust simulation platform for testing would be of great convenience. SnakeSIM is a great base here, but comprehensive work should be put into generalizing it for a more seamless adaptation to different simulation scenarios. A detailed user guide and documentation of the platform would also vastly benefit the research community.

Bibliography

[1] Ø. Stavdahl. "Working note: Hybrid Position/Force Control for Perception Driven Obstacle Aided Locomotion (HOAL) in Snake Robots". Unpublished. 2019.

[2] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl. *Snake robots: modelling, mechatronics, and control*. Springer Science & Business Media, 2012.

[3] P. Liljeback, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl. "Snake robot locomotion in environments with obstacles". In: *IEEE/ASME Transactions on Mechatronics* 17.6 (2011), pp. 1158–1169.

[4] T. Wang, J. Whitman, M. Travers, and H. Choset. "Directional Compliance in Obstacle-Aided Navigation for Snake Robots". In: *arXiv preprint arXiv:2003.01774* (2020).

[5] A. A. Transeth, R. I. Leine, C. Glocker, K. Y. Pettersen, and P. Liljebäck. "Snake robot obstacle-aided locomotion: Modeling, simulations, and experiments". In: *IEEE Transactions on Robotics* 24.1 (2008), pp. 88–104.

[6] T. Klafstad. "Hybrid Position/Force Control for Obstacle Aided Locomotion in Snake Robots". Unpublished. 2019.

[7] A. Koushan. "Simulator for Obstacle Aided Locomotion in Snake Robots". Unpublished. 2019.

[8] H. West and H. Asada. "A method for the design of hybrid position/force controllers for manipulators constrained by contact with the environment". In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE. 1985, pp. 251–259.

[9] M. H. Raibert, J. J. Craig, et al. "Hybrid position/force control of manipulators". In: *Journal of Dynamic Systems, Measurement, and Control* 103.2 (1981), pp. 126–133.

[10] T. Yoshikawa. "Dynamic hybrid position/force control of robot manipulators – description of hand constraints and calculation of joint driving force". In: *IEEE Journal on Robotics and Automation* 3.5 (1987), pp. 386–392.

[11] S. Nansai, M. R. Elara, and M. Iwase. "Dynamic Hybrid Position Force Control using Virtual Internal Model to realize a cutting task by a snake-like robot". In: *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. IEEE. 2016, pp. 151–156.

[12] K. M. Lynch and F. C. Park. *Modern Robotics*. Cambridge University Press, 2017, pp. 272–286.

[13] K. M. Lynch and F. C. Park. *Modern Robotics*. Cambridge University Press, 2017, pp. 421–429.

[14] K. J. Waldron and J. Schmiedeler. "Kinematics". In: *Springer Handbook of Robotics*. Springer, 2016, pp. 11–36.

[15] F. Sanfilippo, J. Azpiazu, G. Marafioti, A. A. Transeth, Ø. Stavdahl, and P. Liljebäck. "Perception-driven obstacle-aided locomotion for snake robots: the state of the art, challenges and possibilities". In: *Applied Sciences* 7.4 (2017), p. 336.

[16] A. A. Transeth and K. Y. Pettersen. "Developments in snake robot modeling and locomotion". In: *2006 9th International Conference on Control, Automation, Robotics and Vision*. IEEE. 2006, pp. 1–8.

[17] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl. "A review on modelling, implementation, and control of snake robots". In: *Robotics and Autonomous Systems* 60.1 (2012), pp. 29–40.

[18] A. J. Ijspeert. "Central pattern generators for locomotion control in animals and robots: a review". In: *Neural networks* 21.4 (2008), pp. 642–653.

[19] C. Holden, Ø. Stavdahl, and J. T. Gravdahl. "Optimal dynamic force mapping for obstacle-aided locomotion in 2D snake robots". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 321–328.

[20] Z. Y. Bayraktaroglu and P. Blazevic. "Understanding snakelike locomotion through a novel push-point approach". In: (2004).

[21] F. Sanfilippo, Ø. Stavdahl, and P. Liljebäck. "SnakeSIM: A ROS-based rapid-prototyping framework for perception-driven obstacle-aided locomotion of snake robots". In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2017, pp. 1226–1231.

[22] A. Calanca, R. Muradore, and P. Fiorini. "A review of algorithms for compliant control of stiff and fixed-compliance robots". In: *IEEE/ASME Transactions on Mechatronics* 21.2 (2015), pp. 613–624.

[23] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl. "Compliant control of the body shape of snake robots". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 4548–4555.

[24] E. H. Moore. "On the reciprocal of the general algebraic matrix". In: *Bull. Am. Math. Soc.* 26 (1920), pp. 394–395.

[25] R. Penrose. "A generalized inverse for matrices". In: *Mathematical proceedings of the Cambridge philosophical society*. Vol. 51. 3. Cambridge University Press. 1955, pp. 406–413.

[26] R. M. Murray. *A mathematical introduction to robotic manipulation*. CRC press, 2017, pp. 155–189.

[27] E. Rezapour, K. Y. Pettersen, P. Liljebäck, J. T. Gravdahl, and E. Kelasidi. "Path following control of planar snake robots using virtual holonomic constraints: theory and experiments". In: *Robotics and biomimetics* 1.1 (2014), p. 3.

[28] S. Chiaverini, G. Oriolo, and I. D. Walker. "Kinematically redundant manipulators". In: *Springer handbook of robotics* (2008), pp. 245–268.

[29] H. Arai, S. Tachi, et al. "Position control of manipulator with passive joints using dynamic coupling". In: *IEEE transactions on Robotics and Automation* 7.4 (1991), pp. 528–534.

[30] A. A. Transeth, N. van de Wouw, A. Pavlov, J. P. Hespanha, and K. Y. Pettersen. "Tracking control for snake robot joints". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2007, pp. 3539–3546.

[31] G. Robinson and J. B. C. Davies. "Continuum robots-a state of the art". In: *Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C)*. Vol. 4. IEEE. 1999, pp. 2849–2854.

[32] M. Kline. *Calculus: an intuitive and physical approach*. Courier Corporation, 1998.

[33] F. Sanfilippo, Ø. Stavdahl, and P. Liljebäck. "SnakeSIM: a ROS-based control and simulation framework for perception-driven obstacle-aided locomotion of snake robots". In: *Artificial Life and Robotics* 23.4 (2018), pp. 449–458.

[34] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[35] N. Koenig and A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. IEEE. 2004, pp. 2149–2154.

[36] Open Source Robotics Foundation. *Tutorial: Using a URDF in Gazebo*. 2014. URL: http://gazebosim.org/tutorials/?tut=ros_urdf (visited on 04/18/2020).

[37] P. Liljebäck, Ø. Stavdahl, K. Y. Pettersen, and J. T. Gravdahl. "Mamba-A waterproof snake robot with tactile sensing". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 294–301.

[38] The MathWorks Inc. *Symbolic Math Toolbox*. Natick, Massachusetts, United State, 2019. URL: https://www.mathworks.com/help/symbolic/.

[39] A. Guillaume. "User and Simulation Interface for Snake Robot's Perception-Driven Obstacle-Aided Locomotion". Unpublished. 2016.

[40] C. Holden and Ø. Stavdahl. "Optimal static propulsive force for obstacle-aided locomotion in snake robots". In: *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2013, pp. 1125–1130.

Atussa Koushan

Hybrid Obstacle Aided Locomotion in Snake Robots

NTNU
Norwegian University of
Science and Technology