NTNU

Norwegian University of
Science and Technology

Path Following and Collision Avoidance for an Underwater Swimming Manipulator

Bjørn Håvard Hoffmann

Master of Science in Cybernetics and Robotics
Submission date: June 2018
Supervisor: Kristin Ytterstad Pettersen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

Increasing interests in subsea operations call for new and improved unmanned underwater vehicles that can solve their tasks while being cost- and energy efficient as well as highly maneuverable. The Underwater Swimming Manipulator (USM) is a robotic system that shows potential in this regard. The system is a hybrid between the biologically inspired Underwater Snake Robot (USR), a manipulator arm, and the more classical Remote Operated Vehicle (ROV) and Autonomous Underwater Vehicle (AUV). The USM can be viewed as a multi-jointed free-floating manipulator with mounted thrusters, which can move by using snake-like motion patterns, by thruster propulsion, or a combination of the two. The robot's multiarticulate body makes it redundant with respect to many of the tasks it needs to perform, as is the case for most multi-jointed manipulators. In addition, the thrusters provide even more freedom to solve a given task. This flexibility is advantageous as it gives the robot excellent maneuverability, but at the same time, the hyper-redundancy makes for a challenging control problem.

To make the USM fully autonomous controllers are needed for a wide range of tasks. One of the most essential of these is for the robot to be able to move around within its working environment, while not colliding with anything. The goal of this project is to develop a path following and collision avoidance system that achieves this for the novel USM.

1. Do a literature survey on path following and collision avoidance of unmanned underwater vehicles, with a focus on hyper-redundant mechanisms.
2. Develop a control system for path following.
3. Develop a control system for collision avoidance.
4. Implement the control systems in MATLAB/Simulink.
5. Validate the control systems through simulations using Vortex Studio.

Abstract

The *Underwater Swimming Manipulator* (USM) is a new type of autonomous unmanned underwater vehicle that combines the highly maneuverable design of the biologically inspired *underwater snake robot* (USR) with thruster actuators as typically used in more traditional underwater vehicle designs. The USM resembles a submerged multiarticulate manipulator arm with thrusters, and can use joint actuation, thruster actuation, or a combination of the two for motion. The thrusters increase the USM's controllable degrees of freedom (DOF), and by placing them strategically they can facilitate 6-DOF rigid-body movement almost independently of the robot's body shape. Together with the long and slender body, this maneuverability enables the USM to gain access to almost any area, that being the inside of shipwrecks or between the steel beams of subsea structure protection cages, and can be utilized to solve complex tasks. One of the most essential abilities of an autonomous mobile robot system is the ability to safely navigate the environment. This thesis proposes controllers for path following and collision avoidance that grants the USM this capability. Both controllers take advantage of the USM's unique design, with the path following controller using the flexible body to improve path following accuracy and reduce energy consumption, while the collision avoidance controller exploits the maneuverability of the robot to enable movement in highly cluttered areas.

Sammendrag

Den *Svømmende undervannsmanipulatoren* (USM) er ny type autonom ubemannet undervannsfarkost som kombinerer det svært manøvrerbare designet til den biologisk inspirerte *undervanns slangeroboten* (USR) med propellere som ofte brukes som framdriftsmiddel på mer tradisjonelle undervannsfarkoster. USMen ligner en flerleddet manipulatorarm med monterte propellere, og kan bruke leddbevegelse, propellkraft eller en kombinasjon av de to for å bevege seg. Propellene øker den styrbare frihetsgraden (DOF) til USMen, og hvis de plasseres strategisk gir de roboten evnen til bevegelse i seks dimensjoner nesten uavhengig av formen kroppen befinner seg i. Sammen med den lange og slanke formen til USMens kropp gir denne manøvrerbarheten roboten tilgang til de fleste områder, det være seg innsiden av et skipsvrak eller mellom stålbjelkene til en subsea struktur, og kan bruke til å løse komplekse oppgaver. En av de mest grunnleggende egenskapene til autonome robotsystemer er deres evne til å navigere området de opererer i. Denne oppgaven presenterer en banefølgings- og en kollisjonsunngåelseskontroller som gir USMen denne ferdigheten. Begge kontrollerene drar nytte av robotens unike design, hvor banefølgingskontrolleren bruker kroppsfleksibiliteten til å forbedre presisjonen og redusere energiforbruket, mens kollisjonsunngåelseskontrolleren utnytter den ekstreme manøvrerbarheten slik at USMen kan bevege seg i svært trange områder.

Preface

This is the concluding work of my Master of Science degree in Engineering Cybernetics at The Norwegian University of Science and Technology. I would like to take the opportunity to thank my supervisor, Prof. Kristin Ytterstad Pettersen, for her invaluable guidance, and for allowing me to take the work in the direction I found most interesting. I would also like to express my gratitude towards my co-supervisor, Jørgen Sverdrup-Thygeson, for supplying me with the necessary material, and for the fruitful discussions on the topics of path following and collision avoidance for the USM. Lastly, I would like to thank my fellow students and friends for five great years at NTNU.

Information detailing background materials and help received for this thesis can be found in chapter 1 under the section *Background*.

Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

AUV	Autonomous Underwater Vehicle
CA	Collision Avoidance
CAP	Collision Avoidance Point
CC	Configuration Controller
COM	Center Of Mass
DOF	Degrees Of Freedom
I-AUV	Intervention Autonomous Underwater Vehicle
IMR	Inspection, Maintenance and Repairs
LOS	Line-Of-Sight
MCF/MCS	Motion Control Framework/System
ROV	Remotely Operated Vehicle
USM	Underwater Swimming Manipulator
USR	Underwater Snake Robot

WLOS Waypoint Line-Of-Sight

Symbols

n Number of joints

n_t Number of thrusters

$\mathcal{F}_i$ The USM joint frame i

$\mathcal{F}_I$ Inertial frame

$\mathcal{F}_b$ The USM base frame

$\mathcal{F}_e$ The USM end-effector frame

$\mathcal{F}_r$ The USM rear-end frame

$\mathcal{F}_{t_j}$ The USM thruster frame i

T_a^c The homogeneous transformation matrix describing the pose of frame $\mathcal{F}_a$ with respect to (wrt.) the frame $\mathcal{F}_c$

R_a^c Rotation matrix describing the orientation of frame $\mathcal{F}_a$ wrt. frame $\mathcal{F}_c$

$\boldsymbol{\Theta_{ac}} = [\phi \quad \theta \quad \psi]^T$ Orientation of frame $\mathcal{F}_c$ wrt. frame $\mathcal{F}_a$ described using Euler angles

$\boldsymbol{p_{ac}^a} = [x_c \quad y_c \quad z_c]^T$ Position of frame $\mathcal{F}_c$ wrt. frame $\mathcal{F}_a$ described using the coordinates of frame $\mathcal{F}_a$

$\boldsymbol{\eta_{ac}^a} = [\boldsymbol{p_{ac}^a}^T \quad \boldsymbol{\Theta_{ac}}^T]^T$ Pose of $\mathcal{F}_c$ relative to $\mathcal{F}_a$ described in $\mathcal{F}_a$

$\boldsymbol{v_{ac}^c} = [u_c \quad v_c \quad w_c]^T \in \mathbb{R}^3$ Linear velocities of $\mathcal{F}_c$ wrt. $\mathcal{F}_a$ described in the body-fixed frame $\mathcal{F}_c$

$\boldsymbol{\omega_{ac}^c} = [p_c \quad q_c \quad r_c]^T \in \mathbb{R}^3$ Angular velocities of $\mathcal{F}_c$ wrt. $\mathcal{F}_a$

$\boldsymbol{q} \in \mathbb{R}^n$ The USM joint angles

$\dot{\boldsymbol{q}} \in \mathbb{R}^n$ The USM joint velocities

$\boldsymbol{V_{ac}^c} = [\boldsymbol{v_{ac}^c}^T \quad \boldsymbol{\omega_{ac}^c}^T]^T \in \mathbb{R}^6$ Velocity of $\mathcal{F}_c$ wrt. $\mathcal{F}_a$

$\boldsymbol{\zeta} = [\boldsymbol{V_{ac}^c}^T \quad \dot{\boldsymbol{q}}^T]^T \in \mathbb{R}^{6+n}$ System velocities

$\hat{V_{ac}^c} \in \mathbb{R}^{4\times4}$ Matrix form (velocity twist) of the velocity vector $\boldsymbol{V_{ac}^c}$

$Ad_{T_a^c} \in \mathbb{R}^{4\times4}$ Adjoint map for the transformation T_a^c

$J_{g,i} \in \mathbb{R}^{6\times(6+n)}$ Geometric Jacobian describing the relationship between the velocities of frame $\mathcal{F}_i$ and the system velocities ζ.

σ Task variable

J_T Task Jacobian

$J_T^\dagger$ Pseudo-inverse

$\boldsymbol{\tau_c} \in \mathbb{R}^6$ Commanded forces and torques

$\tilde{x} = x_{ref} - x$ The tilde represents error variables

$\boldsymbol{u_{thr}} \in \mathbb{R}^{n_t}$ Vector containing the thrust magnitude for each separate thruster

$\boldsymbol{d}$ Shortest vector from the USM's base frame to a line in space

$\boldsymbol{l}$ Vector describing a line in space

$\boldsymbol{v}_{LOS}$ Line-of-sight vector

δ Look-ahead distance

A_c Cross-sectional area of an object

$\boldsymbol{q}_t$ Unit quaternion

c_f Curve factor

$\tau_{CA,i}$ Collision avoidance task for CAP i

$\sigma_{CA,i}$	Collision avoidance task variable for CAP i
$R_{o,i}$	Minimum desired distance to obstacles or CAP i
$\boldsymbol{v_{n,i}}$	Vector from CAP i to the closest obstacle
$\boldsymbol{v_{d,i}}$	Vector from CAP i to a desired position

Chapter 1

Introduction

This chapter will present the motivation behind this thesis and make the reader better acquainted with the problem to be solved. Some parts of the complete control system presented here are the result of previous research, and the distinction between this material and that which belongs to the thesis will be made clear. A short literature review is also given, where some existing knowledge relevant to the thesis' contents is presented.

1.1 Motivation

The ocean is a great resource that has been used by humans throughout history as a means of transportation and as a source of food. In more recent days the ocean is more relevant than ever, as we keep finding new ways to harvest its potential. One of the most important technological contributions that have enabled us to make new advancements on the ocean frontier has been the introduction of unmanned underwater vehicles, as they have significantly improved our ability to go under the surface, and in doing this help us solve a wide variety of tasks. These range from scientific studies of marine lifeforms, mapping of the seafloor and monitoring the impact of global warming to searching for shipwrecks, removing underwater mines

Figure 1.1: Left: HUGIN AUV Right: Oceaneering ROV

and doing inspection and maintenance work on subsea oil installations. This diversity in tasks and operational requirements has resulted in the development of a great number of different unmanned underwater vehicles, both in shape and size. The vehicles are often classified based on how they are operated, and the most common of these classifications are remotely operated underwater vehicles (ROVs), autonomous underwater vehicles (AUVs) and intervention autonomous underwater vehicles (I-AUVs). The ROVs are, as the name implies, remotely controlled by an operator, often through a tether connecting the robot with a surface ship. The AUVs, on the other hand, are autonomous, usually working entirely on their own following predetermined mission objectives or receiving high-level instructions during an operation from a remote location.

One of the fields where unmanned underwater vehicles have proven to be very useful is in the Norwegian oil industry. Extracting oil from the Norwegian continental shelf requires many underwater operations, an in recent years there has been an increasing focus on subsea installations where much of, or the entire, production is moved underwater. This places new demands on the technology that is being used and its capabilities. With production completely at the seafloor, many tasks that were earlier done above the surface now has to be performed subsea, and several of these tasks then fall into the hands of unmanned underwater vehicles.

Although the traditional ROVs and AUVs have been successful in performing the

required subsea tasks until now, the industry is starting to see the need for a new generation of unmanned underwater vehicles that are more cost efficient and able to solve the progressively more complicated tasks that arise. The different tasks needed doing are largely inspection, maintenance, and repairs (IMR). For inspection, AUVs such as the HUGIN (figure 1.1) are suitable, as they are often energy efficient and can operate autonomously for long periods of time. Unfortunately, given their lack of manipulators, they are not able to do any intervention tasks. There does exist AUVs with manipulators, called intervention AUVs (I-AUVs), but they are often large and have poor maneuverability, making them non-ideal for accessing the tight and enclosed areas of subsea installations.

Here the ROVs are much better, as they are generally smaller, and have manipulators and other tools needed to solve the tasks at hand. Still, their use is not without problems. ROVs require a human operator that must be present on a surface ship during the operation, which means large costs and long reaction times. Considering the weaknesses of current day AUVs and ROVs, one could say that there is room for a new kind of unmanned underwater vehicle that combines the autonomy and energy efficiency of the AUV, with the maneuverability and intervening capabilities of the ROV. The *Underwater Swimming Manipulator* (USM) is an unmanned underwater vehicle design that tries to achieve just this.

1.1.1 The Underwater Swimming Manipulator

The USM is in general a robot consisting of a set of links that are connected together by joints, similar to a standard multi-articulated manipulator arm. The links are commonly cylinder-shaped and the joints revolute, but this is no requirement. What separates the USM from a manipulator arm, however, is that it has a number of thrusters mounted on its body. These thrusters can be placed both in the longitudinal and transversal direction of the body. Figure 1.2 shows an example of a USM where the different components can be seen.

The design is a mix between the biologically inspired underwater snake robot (USR) and the more traditional AUV and ROV designs. By giving the USM a long,

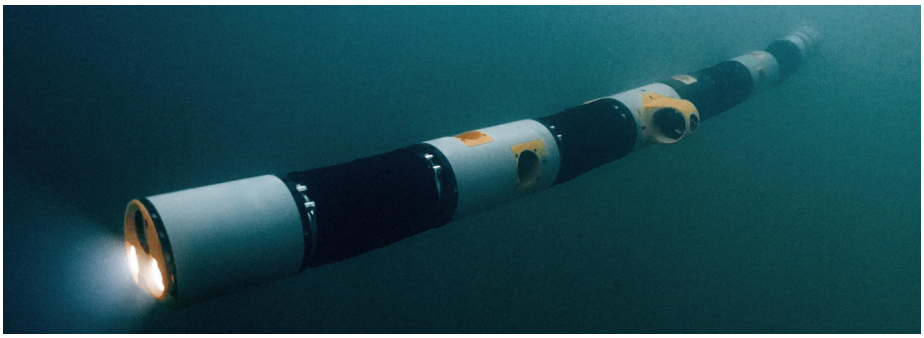


Figure 1.2: The company Eelume's USM. The links are the white modules, the joints are hidden inside the black isolation material that can be seen between the links, and longitudinal thrusters on the sides and transversal thrusters in tunnels can be seen in orange.

slender and multi-articulated body it becomes highly maneuverable in a fashion similar to real-life snakes. The robot's ability to change its body shape is important, and the shape of the USM will often be referred to as its *configuration*. A straight body is an example of one configuration, while an s-shaped body is an example of a different one. Both the form and flexibility of the robot gives it the ability to reach almost any area, that being the inside of a pipe or the tight space between the steel beams of a structural protection cage.

Although it is possible for the USM to navigate by mimicking snake motion patterns alone, this solution does have its limitations. Navigating tight areas becomes difficult, as the entire body must move to generate propulsive forces. Tasks such as dynamic positioning and adjusting for underwater currents also becomes much harder when the robot can only swim. That is why the slender and flexible snake design is combined with traditional naval thrusters.

The thrusters allow for motion of the robot as a rigid body in six degrees of freedom and opens up for a whole new range of maneuvers. If the task is to inspect a pipeline over a long distance the USM can simply straighten its body and use the thrusters to propel itself forward, obtaining a shape and properties similar to the submarine-shaped

AUVs. When navigating enclosed areas it can use its flexible body to twist and avoid obstacles, while using the thrusters for motion. If a tool is mounted at the end of the USM, the thrusters can be used to keep the USM stationed while the body is used for manipulations, giving the USM similar functionality as a mounted manipulator arm.

The flexible and maneuverable design of the USM is promising, and could potentially offer a solution that can replace many of the AUVs and ROVs used for inspection and light maintenance tasks today. However, many problems of both theoretical and practical nature must be solved before such a robot can be realized. This thesis addresses some of these by considering guidance and control of such a robot.

1.2 Problem Description

The USM is intended to be an autonomous vehicle. For the robot to be autonomous, control algorithms must be developed for many different tasks, ranging from the basic task of environment navigation to more specialized tasks like valve turning and docking. Since the tasks are so different in nature the design of the resulting control systems will inevitably be very different. However, common for them all is that if they are to solve the tasks effectively they should all take the unique design of the USM into consideration.

The multi-articulated body and many thrusters give the robot a large degree of freedom, and can potentially be used to design effective control algorithms enabling the robot to solve complex tasks. At the same time, this freedom is part of what makes the control problem challenging, as there is no trivial way to resolve how to best actuate the robot for it to achieve its goal at any given time. Should the robot move by actuating its joints, should it use only the thrusters, or is the optimal solution a combination of the two?

One of the most essential abilities of an autonomous mobile robot is its ability to navigate its working environment. This is a complex problem that involves many components: Sensors are needed to gather information about the environment, and this data must be processed and used to build a map of the robot's surroundings. Based on the map a motion planning system can generate paths for the robot to follow. A path

following controller is needed to guide the robot along the desired path, and finally, a motion controller is responsible for actually actuating the robot so that the desired motion takes place. It is also often necessary with a collision avoidance controller that makes sure that the robot does not collide with any obstacles when moving about its environment, as collisions can lead to damage on both the robot itself and the object it collides with.

This thesis considers the problem of navigation for the novel underwater swimming manipulator and presents controllers for path following and collision avoidance as part of the solution to this task. The controllers are designed to work in 3D, and have been implemented using Simulink/Matlab and tested using the simulation environment Vortex.

1.3 Background

Path following and collision avoidance are tasks that are part of what is often called the robot's *guidance system*. The guidance system is responsible for controlling the general motion of the robot, similarly to how the human mind decides on what the body should do. The decisions made by the guidance system are parameterized as some reference variables that are then sent to what is commonly called the motion control system. The motion control system's job is to carry out the orders given by the guidance system by actually actuating the robot. In the human analogy, this is the same as using our legs to actually cross the room after we have decided that is what we want to do. Guidance laws can be discussed in a general sense without reference to a motion control system, but what kind of motion that is feasible for a specific robotic system is governed by its physical shape and actuators, and the motion control system's ability to utilize them. It thus makes more sense to discuss guidance systems in the context of the robot and the motion control system they will be used with. Furthermore, if the qualities of a guidance system are to be tested, either through simulations or by real-world experiments, a motion control system is a prerequisite.

Ph.D. candidate Jørgen Sverdrup-Thygeson at the Institute of Engineering Cybernetics at NTNU has through his research looked at ways to effectively control a USM.

As a result, he has designed a framework for motion control of USMs. In addition to having presented the theoretical aspects of the framework in his paper [31], he has also implemented it in MATLAB/Simulink code. Furthermore, he has created a full 3D simulation model of a USM in the simulation software Vortex, with which the motion control framework can communicate. Together the MATLAB/Simulink motion control framework and the Vortex simulation model enables the development and testing of control algorithms through simulations. The framework and simulation model has kindly been provided to the author for the work in this thesis, which would not have been possible otherwise.

The details of the motion control framework (MCF) supplied by Jørgen Sverdrup-Thygeson will be discussed in chapter 3. However, in order to better fit with the guidance system developed in this thesis some parts of the MCF has been changed, and thus the MCF will be briefly introduced in order to highlight these changes. The framework written in MATLAB/Simulink consists of four parts: The Kinematic controller, the dynamic controller, the thruster allocation module and the interface that is used to communicate with the Vortex simulation software. The thruster allocation module has been left untouched and is completely as provided by Sverdrup-Thygeson. The forward kinematics in the kinematic controller has been modified, as different body-fixed reference frames are used than in Sverdrup-Thygeson's work. The inverse kinematics in the kinematic controller has been completely rewritten, so as to be compatible with the path following and collision avoidance guidance laws. A new dynamic controller has also been created, albeit based on the linearization feedback controller suggested in Sverdrup-Thygeson's work [31]. Finally, the interface has been extended to receive and process extra data from the Vortex simulation software, as an additional measurement point has been added and one of the previously available ones has been moved. The Vortex simulation model itself is unaltered.

1.4 Contribution

The main contributions of this thesis are the path following and collision avoidance controllers developed for the novel underwater swimming manipulator. These are two

essential properties of an autonomous system and often serves as a basis for control systems that perform more advanced tasks. Path following and collision avoidance for the USM in a complete 3D environment has not been done before this thesis. The different techniques employed to achieve path following and collision avoidance are, when considered separately, not novel in nature. However, the USM is a robotic system unlike most other, and the combination of methods used in this thesis has, to the author's best knowledge, not been used before.

1.5 Literature Review

The underwater swimming manipulator has traits that are similar to those of both multiarticulate manipulator arms and traditional underwater vehicles. Manipulator arms and underwater vehicles have been topics of research for many decades, and it is thus natural to look into literature from these fields for inspiration when designing control systems for the USM. When dealing with mobile robot systems collision avoidance is also often of importance, as the robot system often navigates in cluttered environments. This section will present some earlier research on these three topics, that is relevant to the work presented this thesis.

1.5.1 Inverse Kinematics

Multiarticulate robotic arms have been used in the industry and for research purposes for a long time, and as both the mechanisms themselves and their control algorithms keep evolving their usage is ever more widespread. When considering control of manipulator arms, kinematics and inverse kinematics are essential. As stated in [29], the direct kinematic equations establish the functional relationship between the joint variables and the end-effector position and orientation. The *inverse kinematics problem* (IK problem) consists of the determination of the joint variables corresponding to a given end-effector position and orientation. For most control problems it is desirable to specify the task through end-effector position and orientation, velocity or acceleration. Considering this the solution to the inverse kinematics problem is of fundamental

importance.

There exist many alternatives for trying to solve the IK problem [9], such as interval methods [27], distance-based methods [26], neural networks [10], Bézier maps [34] and, probably the most popular method, closed-loop inverse kinematics (CLIK) algorithms [20]. The CLIK algorithms use a first-order Jacobian matrix [24] of the robot kinematics that maps joint velocities into task space velocities, and inverts this matrix to find the joint velocities that reduce the task error. One major advantage of the CLIK algorithms is that they do not require any previous knowledge or learning process with the robot, only the Jacobian matrix. The algorithms vary mainly in how they choose to compute the inverse of the Jacobian, and how they deal with problems related to singularities. A comprehensive list of the different CLIK methods can be found in [9].

Redundant manipulators are manipulators which have more degrees of freedom than are needed to solve the given task. When this is the case it is often desirable to use the extra degrees of freedom to solve some secondary tasks. This is commonly done by projecting the gradient of a secondary task through the kernel of the Jacobian matrix, so as not to affect the primary task [23]. This method has been extended to handle both singularity issues [7] and multiple tasks [30], resulting in it being known as the Singularity-Robust Multiple Task Priority (SRMTP) method. In [2] set-based control is incorporated with the SRMTP method.

1.5.2 Path Following

Path following can be defined as the act of following a predefined path in space independent of time. The same problem but with a temporal constraint is called trajectory tracking. For both path following and trajectory tracking it is possible to add spacial constraints which represent obstacles and other positional constraints, should they be known in advance. To successfully achieve path following two things are needed, a guidance law that determines how the vehicle should move and a motion controller that makes the vehicle move accordingly.

The subject of guidance originated within the guided missile community, but its principles are today used in many different applications where motion control is

needed. An overview of several classical guidance laws for both target tracking and path following can be found in [4]. One of the most widely used methods for guidance in path following controllers is the line-of-sight (LOS) guidance law (ch. 10, [12]). Some reasons for the method's popularity are that it is simple, intuitive, easy to tune and, in the case of no environmental disturbances, provides nice path convergence properties. Although the LOS guidance law, in theory, works for paths of any shape, it is mainly used for straight line paths or paths with simple geometries. The LOS method has also been extended with an integral effect to counter external disturbances like ocean currents and wind [17], [3]. For paths of arbitrary shape, the Serret-Frenet equations are often used as a guidance law [4], which has also seen extensions that handle external disturbances [21].

1.5.3 Collision Avoidance

For most mobile robot systems it is desirable to avoid collisions with any obstacles in the environment, as this can lead to damage to both the robot and the environment. This process of avoiding obstacles is called collision avoidance and has been relevant for as long as there have been mobile robots. In the literature, there are in general two different approaches to collision avoidance, reactive algorithms, and deliberate algorithms.

Reactive collision avoidance algorithms make decisions based on the current situation only, which enables them to react quickly to rapid changes in the environment. However, the solutions they produce are often sub-optimal because the resulting future state of the system is not taken into consideration. The reactive algorithms are often implemented by using some guidance controller that steps in and steers the robot safely away from an obstacle when needed [22]. Some examples of reactive algorithms include velocity obstacles [11] and the dynamic window approach [13].

Deliberate algorithms use all available information accumulated about the environment to make decisions that are optimal in a global sense. These algorithms are thus often part of the motion planning system that generates the robots desired path. Some examples of deliberate algorithms are the rapidly-exploring random trees [19],

graph search algorithms such as A* [25] and constrained nonlinear optimization and model predictive control [16]. The disadvantage of the deliberate algorithms is that they are computationally heavy, and thus often not feasible for real-time usage.

To exploit their strengths and minimize their weaknesses reactive and deliberate algorithms can be used together in a hierarchical structure. Together with other motion control functions like path following, the complete system is then able to conduct long-term planning to find optimal paths and react to rapid changes in the environment or unpredicted events.

1.6 Outline

The rest of this thesis is structured in the following way: First, a kinematic model for the USM is presented in chapter 2. Then the previously mentioned motion control framework will be discussed in chapter 3. Chapter 4 introduces the path following and collision avoidance controllers developed in this thesis, followed by some implementation details and simulation results demonstrating the controllers' capabilities in chapter 5. Finally, chapter 6 brings the thesis to an end with conclusions and some suggestions for future work.

Chapter 2

Kinematic Model

In this chapter, a kinematic model for a general USM will be presented. The model is general, as it is valid for an arbitrary number of links, joints, and thrusters, and the different components can be placed freely, assuming that the general structure of a USM is kept. The model that is presented is a recollection of the model given in [33], with some adjustments to notation. In addition, the differential kinematics of the model will be derived.

2.1 Definitions and Notation

The multiarticulate body of the USM gives it properties similar to a manipulator arm. Manipulator arms are mounted to a structure called a base, which can be both mobile or stationary. By imagining a point on the USM as its base, and the rest of the robot as a manipulator arm, one can adopt the same view for the USM. This view of the robot, as a floating manipulator arm, enables the use of control methods used for manipulator arms. The base also serves as a reference point on the robot for the position and orientation (pose) of the rest of its body in space. For instance, if the base is chosen to be the center of mass (COM) of the rearmost link of the USM, this link can be seen as a floating base and the rest of the robot a manipulator arm. If, on the other hand, the

base is chosen to be the COM of the center link of the USM, this link can be seen as a floating base with two separate manipulator arms, one on each side. This difference is also reflected in the kinematic model, as it depends on the choice of base.

2.1.1 Reference Frames

The USM consists of $n+1$ links, connected by n actuated joints (see figure 2.1). Each link is numbered from the tail and forward, indicated by the subscript $i \in [0 \dots n]$. Each link has a link reference frame attached to its center of mass, denoted $\bar{\mathcal{F}}_i$, with axes that coincide with the principal axes of inertia of the link. A frame for the chosen base of the USM is defined as $\mathcal{F}_b$. In addition to the base frame, frames are defined for other points of interest on the USM as well. One such point is the end-effector frame, $\mathcal{F}_e$, located at the very end of the frontmost link, where one would likely mount a tool or a camera. The joints have one degree of freedom and are numbered $i \in [1 \dots n]$ such that the links i and $i-1$ are connected by joint i. The corresponding joint reference frames are fixed to the center of the joints and denoted $\mathcal{F}_i$. The USM can be equipped with both longitudinal thrusters along its sides and transversal tunnel thrusters through the links. There are a total of m thrusters that are numbered by $j \in [1 \dots m]$. Their position and orientation is described by the reference frames $\mathcal{F}_{t_j}$, where the thruster force point of application is at the origin of the frame and the thrust vector lies along the x-axis of the frame. The chosen inertial reference frame is denoted $\mathcal{F}_I$.

2.1.2 Homogeneous Transformations

The body-fixed reference frames defined in the previous section represents the position and orientation of different parts of the USM in space, and together they represent the entire robot. One way to describe the relationship between the different frames is through homogeneous transformation matrices. As an example, the pose of the base frame $\mathcal{F}_b$ with respect to the inertial frame $\mathcal{F}_I$ is given by the homogeneous transformation matrix

$$T_b^I = \begin{bmatrix} R_b^I & p_{Ib}^I \\ 0 & 1 \end{bmatrix} \in SE(3) \tag{2.1}$$

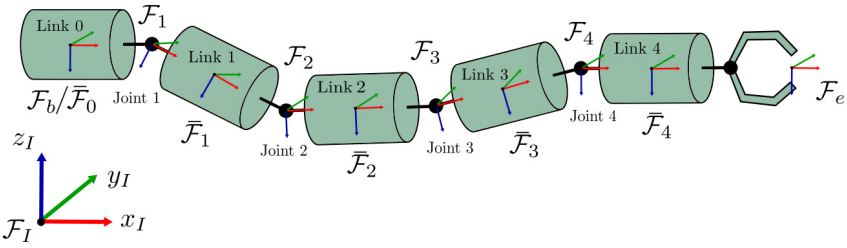


Figure 2.1: Illustration showing the different reference frames used in the kinematic model for a USM with 5 links, $n = 4$ joints, a gripper tool, and with the base defined as the COM of the rearmost link. Thrusters and thruster frames are not visualized.

where $R_b^I \in SO(3)$ is the rotation matrix describing the orientation of the base frame with respect to the inertial frame and p_{Ib}^I is the vector from the origin of $\mathcal{F}_I$ to the origin of $\mathcal{F}_b$ described in the coordinates of frame $\mathcal{F}_I$. Note: For ease of notation the superscript on the transnational vectors will only be used if the vector is described in a coordinate system other than the inertial frame, i.e. $p_{Ib} = p_{Ib}^I$.

2.1.3 Position, Orientation and Velocities

The position and orientation of a frame $\mathcal{F}_c$ in space relative some inertial frame $\mathcal{F}_a$ is described by its pose

$$\boldsymbol{\eta}_{ac}^{a} = [\boldsymbol{p}_{ac}^{a\ T} \quad \boldsymbol{\eta_o}^T]^T \tag{2.2}$$

where $\mathbf{p}_{\mathbf{ac}}^{\mathbf{a}} = [x_c \quad y_c \quad z_c]^T$ are the cartesian coordinates of the origin of frame $\mathcal{F}_c$ with respect to frame $\mathcal{F}_a$ described in frame $\mathcal{F}_a$, and $\boldsymbol{\eta}_{\mathbf{o}}$ describes the orientation of the frame. The contents of $\boldsymbol{\eta}_{\mathbf{o}}$ depends on how the rotation is parameterized. For instance, by using Euler angles $\boldsymbol{\eta}_{\mathbf{o}} \in \mathbb{R}^{3\times1}$, while using unit quaternions would result in $\boldsymbol{\eta}_{\mathbf{o}} \in \mathbb{R}^{4\times1}$. In this thesis Euler angles are used, implying that $\boldsymbol{\eta}_{\mathbf{o}} = [\phi \quad \theta \quad \psi]^T$, representing the frame's rotation about the x, y and z-axis of the inertial frame respectively. In this thesis the name convention found in marine craft literature will be adopted, implying that the Euler angles ϕ, θ, and ψ are called roll, pitch and yaw.

The linear and angular velocities of a frame $\mathcal{F}_c$ with respect to a frame $\mathcal{F}_a$ are described by

$$\boldsymbol{v}_{ac}^{c} = \begin{bmatrix} u_c \\ v_c \\ w_c \end{bmatrix} \in \mathbb{R}^3, \ \boldsymbol{\omega}_{ac}^{c} = \begin{bmatrix} p_c \\ q_c \\ r_c \end{bmatrix} \in \mathbb{R}^3 \tag{2.3}$$

respectively. The linear velocities, along the x, y and z axis, are called *surge*, *sway* and *heave*, while the angular velocities p, q and r are referred to as *roll*, *pitch* and *yaw* rates.

In addition to describing the pose and velocities of independent frames, it is necessary to define variables that describe the system velocity, i.e. the velocities of the USM. By choosing the base frame as a reference frame for the USM, the system velocities can be described by

$$\zeta = \begin{bmatrix} \boldsymbol{V}_{Ib}^{b} \\ \dot{\boldsymbol{q}} \end{bmatrix} \in \mathbb{R}^{6+n}, \ \boldsymbol{V}_{Ib}^{b} = \begin{bmatrix} \boldsymbol{v}_{Ib}^{b} \\ \boldsymbol{\omega}_{Ib}^{b} \end{bmatrix} \tag{2.4}$$

where $\boldsymbol{v}_{Ib}^{b}$ and $\boldsymbol{\omega}_{Ib}^{b}$ are the body-fixed linear and angular velocities of the base of the USM with respect to the inertial frame, as described above, and $\dot{\boldsymbol{q}}$ is the vector of joint velocities.

2.2 Kinematics

2.2.1 Forward Kinematics

In this thesis, it is assumed that the homogeneous transformation between the inertial frame and the base frame, T_b^I, is known. In addition, assuming that the dimensions of the USM and the joint angles q are known, the transformation matrices between joint i and joint $i-1$ and its inverse [14] are known and given as

$$T_i^{i-1}(q_i) = \begin{bmatrix} R_i^{i-1}(q_i) & \boldsymbol{p}_{(i-1)i}^{i-1} \\ 0 & 1 \end{bmatrix}, \ T_i^{i-1^{-1}}(q_i) = \begin{bmatrix} R_i^{i-1^T}(q_i) & -R_i^{i-1^T}(q_i)\boldsymbol{p}_{(i-1)i}^{i-1} \\ 0 & 1 \end{bmatrix} \tag{2.5}$$

Through consecutive transformations the transformation from the inertial frame to any of the frames can be found by

$$T_i^I(q_1 \dots q_i) = T_b^I T_1^b(q_1) T_2^1(q_2) \dots T_i^{i-1}(q_i) = \begin{bmatrix} R_i^I(q_1 \dots q_i) & \boldsymbol{p_{Ii}} \\ 0 & 1 \end{bmatrix} \tag{2.6}$$

These transformation matrices are used to represent the forward kinematics of the USM. It should also be noted that they are time-dependent, as both the base pose and joint angles are time-varying. For notational convenience the explicit dependence on $\boldsymbol{q}$ will often be omitted, i.e. $T_i^I \triangleq T_i^I(q_1 \dots q_i)$.

2.2.2 Differential Kinematics

The forward kinematics provides a mapping from the joint angles to a set of variables that describe the pose of the robot. For instance, the relationship between the joint angles and the pose of the end-effector relative to the base frame can be described by the transformation matrix T_e^b, where information about orientation and position is found within the matrix in R_e^b and $\boldsymbol{p_{be}^b}$. A more general way of describing this mapping is

$$\boldsymbol{\eta_{be}^b} = \boldsymbol{k(q)} \tag{2.7}$$

where the vector $\boldsymbol{\eta_{be}^b}$ is some chosen representation of the USM's end-effector pose, and $\boldsymbol{q}$ are the joint angles. Usually, when designing control algorithms for robots moving in Euclidean space it is desirable to specify control tasks in terms of the robot's pose, not in terms of the joint angles. In order to do this the inverse mapping $\boldsymbol{k^{-1}(q)}$ is needed. Unfortunately the mapping $\boldsymbol{k(q)}$ is in general not invertible. A common solution to this problem is to solve the inverse kinematics at the differential level, giving rise to differential kinematics. The differential kinematics thus describe the relationship between the velocities of a part of the USM, for instance the end-effector, and the joint velocities.

The differential kinematics is found by differentiating the forward kinematics, i.e

the homogeneous transformation matrix, which results in

$$\dot{T}_b^I = \hat{V}_{Ib}^b T_b^I \;\Rightarrow\; \hat{V}_{Ib}^b = {T_b^I}^{-1}\dot{T}_b^I \tag{2.8}$$

where $\hat{V}_{Ib}^b$ is the matrix form of the body-fixed base frame velocities, called the velocity twist of V_{Ib}^b, and given as

$$\hat{V}_{Ib}^b = \begin{bmatrix} \hat{\omega}_{Ib}^b & v_{Ib}^b \\ 0 & 0 \end{bmatrix} \tag{2.9}$$

where $\hat{\omega}_{Ib}^b$ is the skew-symmetric matrix of ω_{Ib}^b.

The velocity twist of a link i with respect to the inertial frame $\mathcal{F}_I$ can then be expressed in frame $\mathcal{F}_i$ by

$$\begin{aligned} \hat{V}_{Ii}^i &= {T_i^I}^{-1}\dot{T}_i^I && (2.10)\\ &= (T_b^I T_i^b)^{-1}(\dot{T}_b^I T_i^b + T_b^I \dot{T}_i^b) && (2.11)\\ &= {T_i^b}^{-1}\hat{V}_{Ib}^b T_i^b + \hat{V}_{bi}^i && (2.12) \end{aligned}$$

This expression can be written in coordinate form as

$$V_{Ii}^i = Ad_{T_i^b}^{-1} V_{Ib}^b + V_{bi}^i \tag{2.13}$$

where $Ad_{T_i^b}^{-1}$ is the adjoint map for the transformation T_i^b [14]. V_{Ii}^i describes the velocity of link i with respect to the inertial frame, described in frame i. For the homogeneous transformation matrix the adjoint map and its inverse are given by

$$Ad_{T_a^b} = \begin{bmatrix} R_a^b & \hat{p}_{ba}^b R_a^b \\ 0 & R_a^b \end{bmatrix},\; Ad_{T_a^b}^{-1} = \begin{bmatrix} {R_a^b}^T & -{R_a^b}^T \hat{p}_{ba}^b \\ 0 & {R_a^b}^T \end{bmatrix} \tag{2.14}$$

By following the same steps as above, the body-fixed velocity of the end-effector frame $\mathcal{F}_e$ is found to be

$$V_{Ie}^e = Ad_{T_e^b}^{-1} V_{Ib}^b + V_{be}^e \tag{2.15}$$

In order to complete the equations 2.13 and 2.15 expressions must be found for V_{bi}^{i} and V_{be}^{e}, the relative link and end-effector velocities with respect to the base. These relative velocities are linked to the joint velocities through a linear mapping by the Jacobian [29]

$$V_{bi}^{i} = Ad_{T_i^b}^{-1} J_i \dot{q} \tag{2.16}$$

$$V_{be}^{e} = Ad_{T_e^b}^{-1} J_e \dot{q} \tag{2.17}$$

where the link Jacobian J_i and the end-effector Jacobian J_e are given by

$$J_i(q) = [Ad_{T_1^b}(q)X_1^1 \dots Ad_{T_i^b}(q)X_i^i,\ 0_{6\times(n-i)}] \tag{2.18}$$

$$J_e(q) = [Ad_{T_1^b}(q)X_1^1 \dots Ad_{T_n^b}(q)X_n^n] \tag{2.19}$$

and X_i^i are the joint twist coordinate vectors given by either $X_i^i = [0,0,0,0,0,1]^T$ or $X_i^i = [0,0,0,0,1,0]^T$ depending on whether the joint rotates about the z-axis or the y-axis of frame $\mathcal{F}_i$. Finally the differential kinematics are found by inserting 2.16 and 2.17 into 2.13 and 2.15:

$$V_{Ii}^{i} = J_{g,i}(q)\zeta, \quad J_{g,i}(q) = [Ad_{T_i^b}^{-1}, Ad_{T_i^b}^{-1} J_i] \in \mathbb{R}^{6\times(6+n)} \tag{2.20}$$

$$V_{Ie}^{e} = J_{g,e}(q)\zeta, \quad J_{g,e}(q) = [Ad_{T_e^b}^{-1}, Ad_{T_e^b}^{-1} J_e] \in \mathbb{R}^{6\times(6+n)} \tag{2.21}$$

$J_{g,i}$ and $J_{g,e}$ are the geometric Jacobians, and maps the body-fixed base velocities and joint velocities to the linear and angular velocities of each link and the end-effector respectively.

Chapter 3

Motion Control System Framework

The goal of this thesis is to develop control systems for path following and collision avoidance for the USM. Both collision avoidance and path following are tasks that are often handled by guidance systems that guide the motion of the robot to achieve the desired behavior. It is the responsibility of the motion control system to actuate the robot to create the desired motion. The motion control system thus directly influences any guidance system, as it defines what kind of maneuvers that are possible. The USM is a complex robot with its many thrusters and multiarticulate body, and designing an effective motion control system that can be used to fully utilize the robot's maneuverability is not trivial.

In the paper *A control framework for biologically inspired underwater swimming manipulators equipped with thrusters* [31] Sverdrup-Thygeson et al. presents a motion control system framework for the USM. This framework has been used as a basis for the motion control system used together with guidance systems to create the path following and collision avoidance controllers presented in this thesis. This chapter will thus introduce the framework and its components. The material in this chapter is

based mainly on the results in the above-mentioned paper, as well as a more recent paper from the same author [14].

3.1 Overview

The motion control framework draws inspiration from typical guidance, control and allocation systems that have previously been used for both underwater and surface vessels. Figure 3.1 shows the components of the motion control system suggested by Sverdrup-Thygeson for the USM.

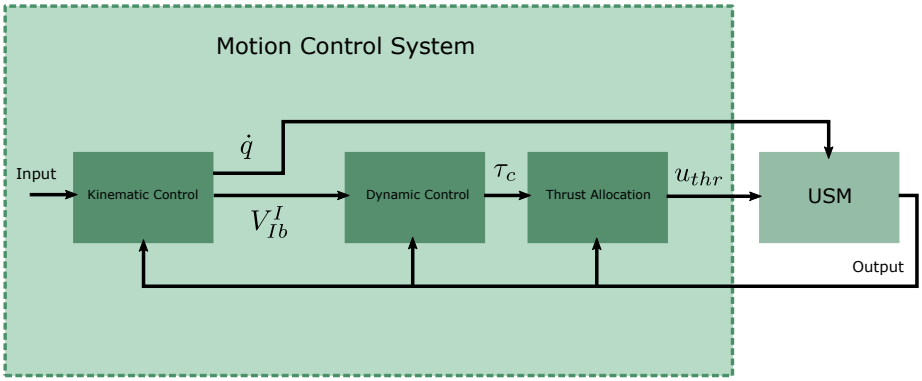


Figure 3.1: Motion Control Framework

The main responsibility of each component is:

- Kinematic Control: Based on a given task the kinematic controller calculates reference joint velocities and reference linear and angular velocities for the base of the USM. A typical task could be to move the end-effector to a desired position.
- Dynamic Control: Takes the reference linear and angular base velocities from the Kinematic Control block and calculates general forces needed to achieve these velocities.

- Thrust Allocation: Calculates the required thruster actuation that is needed from each separate thruster in order to generate forces equal to the reference forces, while taking the robot's configuration into consideration.

- USM: The Underwater Swimming Manipulator represented as the real physical robot or a simulation model.

The following sections will discuss the different components of the motion control system in closer detail.

3.2 Kinematic Control

The kinematics describe the position and orientation of both the entire robot and its individual parts with respect to each other. Kinematic control is thus of much interest, as it can be used to control the shape of the USM's body. Since the USM can be viewed as a floating base manipulator arm it is natural to look at methods developed for control of land-based manipulator arms for inspiration, when determining how to do the kinematic control. A popular control method from this field is the closed-loop inverse kinematics algorithm (CLIK) and is the one that has been adopted in the USM control framework.

3.2.1 Closed-Loop Inverse Kinematics

The forward kinematics of the USM are given by its kinematic model and describes the robot's pose and configuration in space given the joint angles. As mentioned in chapter 2 it is desirable to have the inverse relation, the inverse kinematics, when controlling the robot, such that a desired pose in space can be used as an input, and the required joint angles and base position is given as control references. Such a relationship was found on a velocity level through the differential kinematics and were given by a mapping through the geometric Jacobian (equations 2.20, 2.21).

In addition to having the inverse kinematics for the USM, the control tasks themselves must be defined. Typical control tasks might be controlling the orientation of

the end-effector, the position of the base, or keeping a minimum distance to an obstacle. A more comprehensive list of possible control tasks can be found in [1]. The different tasks are defined by task variables, which in general are given as

$$\sigma = \sigma(\eta, \mathbf{q}) \tag{3.1}$$

where η represents the pose of the robot's base and $\mathbf{q}$ are the joint angles. In same way as for the differential kinematics it is possible to find a Jacobian that relates the task derivative to the system velocity

$$\dot{\sigma} = \mathbf{J_T}(\eta, \mathbf{q})\boldsymbol{\zeta} \tag{3.2}$$

where $\mathbf{J_T}$ is the corresponding task Jacobian.

With the proper task Jacobian, the reference system velocities ζ_r can be found simply by inverting the mapping 3.2. A common way to do this is to use the pseudoinverse of the task Jacobian, resulting in

$$\boldsymbol{\zeta_r} = \mathbf{J_T}^\dagger \dot{\sigma}_d, \quad \mathbf{J_T}^\dagger = \mathbf{J_T^T}(\mathbf{J_T}\mathbf{J_T^T})^{-1} \tag{3.3}$$

where $\dot{\sigma}_d$ are the desired task velocities. The velocities can then be numerically integrated in order to get the reference position and joint angles. Unfortunately the integration often leads to drifting because of numerical errors. To remedy this it is common to instead define an error variable to obtain a closed-loop version of the algorithm:

$$\begin{aligned}
\tilde{\sigma} &= \sigma_d - \sigma \\
\dot{\tilde{\sigma}} &= \dot{\sigma_d} - \mathbf{J_T}\boldsymbol{\zeta} \\
\boldsymbol{\zeta_r} &= \mathbf{J_T}^\dagger(\dot{\sigma_d} - \dot{\tilde{\sigma}})
\end{aligned}$$

The closed-loop inverse kinematics algorithm is found by making a small alteration to the above equation

$$\boldsymbol{\zeta_r} = \mathbf{J_T}^\dagger(\dot{\sigma_d} - K\dot{\tilde{\sigma}}) \tag{3.4}$$

where K is chosen as a positive definite, and often symmetric, matrix to assure asymptotic stability in the error variable [29].

3.2.2 Singularity-Robust Multiple Task Priority

A common property of manipulator arms is redundancy. A manipulator arm is said to be redundant if it has more degrees of freedom than are needed to solve the task at hand. Figure 3.2 shows a redundant manipulator arm in 2D. Here the task is to position the end-effector, a task with a dimensionality of two, while the manipulator arm has three rotational joints, i.e. three degrees of freedom. The result of this extra degree of freedom is that there is an infinite number of solutions to the positioning task. Three possible solutions are shown in the figure 3.2.

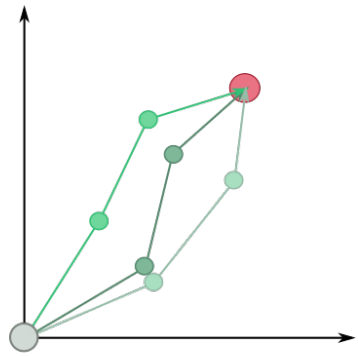


Figure 3.2: Kinematic redundancy in 2D

The extra degrees of freedom in a redundant manipulator can be exploited by defining secondary kinematic tasks for the robot to solve. There are several ways to do this, but in the framework presented here, this is done by using the singularity-robust multiple task priority (SRMTP) method [30]. This leads to an extension of the CLIK

algorithm given in equation 3.4, which now becomes

$$\boldsymbol{\zeta_r} = \mathbf{J_1}^{\dagger}(\dot{\sigma_{1,d}} - K_1\dot{\tilde{\sigma}_1}) + N_1\mathbf{J_2}^{\dagger}(\dot{\sigma_{2,d}} - K_2\dot{\tilde{\sigma}_2}) + \cdots + N_{12..(k-1)}\mathbf{J_k}^{\dagger}(\dot{\sigma_{k,d}} - K_k\dot{\tilde{\sigma}_k}) \quad (3.5)$$

where k is the number of tasks, $\boldsymbol{J_i}$ is the i-th task Jacobian, $\boldsymbol{N_i} = (\mathbf{I} - \mathbf{J_i}\dagger\mathbf{J_i})$ is the null space of $\boldsymbol{J_i}$ and $\boldsymbol{N_{12..(k-1)}}$ is the null space of the combined jacobians, i.e. the null space of $\boldsymbol{J_{12..(k-1)}} = [J_1^T\ J_2^T\ \dots\ J_{k-1}^T]^T$. The prioritization of the tasks is strict and given by their indices, where $i = 1$ is the top prioritized task.

3.3 Dynamic Control

The dynamic controller receives velocity references from the kinematic control module and calculates general forces based on these. In theory, the system designer could choose any controller for this purpose. However, there is currently no 3D analytical model available for the USM, limiting the possible choices. In [33] Sverdrup-Thygeson proposes to use a state-feedback linearization controller

$$\boldsymbol{\tau_c} = M_{11}(q)\boldsymbol{a^b} + n(R_b^I, q, \zeta) \quad (3.6)$$

where τ_c is the vector of commanded generalized forces and moments, $M_{11}(q)$ is part of the system inertia matrix, $n(R_b^I, q, \zeta)$ is the nonlinear feedback linearization matrix and a^b is the commanded body-fixed acceleration vector given by

$$\boldsymbol{a^b} = \dot{V}_{Ib}^b - K_p\tilde{V}_{Ib}^b - K_I\int_0^t \tilde{V}_{Ib}^b d\tau \quad (3.7)$$

The nonlinear feedback linearization matrix is not known, as this requires model knowledge of the system. However, Sverdrup-Thygeson argues that the linearizing term can be ignored when the robot moves at low velocities, as its influence then will be small.

3.4 Thrust Allocation

In order to actually generate the forces calculated by the dynamic controller, the thrusters must be correctly actuated. The pose of each individual thruster relative to the base depends on the USM's configuration, implying that even if the total reference force vector is constant for a period of time, the corresponding thruster actuation will still change if the robot's configuration changes. A scenario like this is depicted in figure 3.3. The USM's actuator forces $\tau(q)$ are described by

$$\tau(q) = \begin{bmatrix} \tau_b(q) \\ \tau_q(q) \end{bmatrix} = \begin{bmatrix} T(q) & 0_{6\times n} \\ 0_{n\times m} & I_{n\times n} \end{bmatrix} \begin{bmatrix} u_{thr} \\ u_q \end{bmatrix} \tag{3.8}$$

there $\tau_b(q)$ are the base forces and torques, $\tau_q(q) = u_q$ are the joint torques, u_{thr} are the thruster forces, and $T(q)$ is the thruster configuration matrix (TCM) that translates the thruster forces into forces and torques acting on the base. In chapter 2 a linear mapping between the link frame and system velocities was found through the geometric Jacobian (2.20). In a similar manner, the geometric Jacobians for the thruster frames that maps the forces and moments from the thruster frames to the base frame can be found, and using these the total force and torque experienced by the base frame can be expressed as

$$\boldsymbol{\tau(q)} = \sum_{j=1}^{m} J_{g,t_j}^T \boldsymbol{\tau_{t_j}} \tag{3.9}$$

where $\tau_{t_j} = [1\ 0_{1\times 5}]^T u_{thr,j}$ and $u_{thr,j}$ is the scalar force applied by thruster j. The base forces and torques $\boldsymbol{\tau_b}$ are those of interest, and can be extracted from $\boldsymbol{\tau}$ by applying the selection matrix $H = [I_{6\times 6}\ 0_{6\times n}]$

$$\boldsymbol{\tau_b(q)} = H\boldsymbol{\tau(q)} = \sum_{j=1}^{m} H J_{g,t_j}^T \boldsymbol{\tau_{t_j}} = \sum_{j=1}^{m} (Ad_{T_{t_j}^b}^{-1})^T \boldsymbol{\tau_{t_j}} = \sum_{j=1}^{m} B_j(q) u_{thr,j} \tag{3.10}$$

where the definition of J_{g,t_j} found in 2.20 has been used, and

$$B_j(q) \triangleq (Ad^{-1}_{T^b_{t_j}})^T \begin{bmatrix} 1 \\ 0_{5\times1} \end{bmatrix} = \begin{bmatrix} {R^b_{t_j}}^T & 0 \\ {\hat{p}^b_{bt_j}}{R^b_{t_j}}^T & {R^b_{t_j}}^T \end{bmatrix} \begin{bmatrix} 1 \\ 0_{5\times1} \end{bmatrix} \tag{3.11}$$

By comparing equations 3.8 and 3.10 it can be seen that the thruster configuration matrix can be expressed as the horizontal concatenation of the column vectors $B_j(q)$

$$T(q) = [B_1\ B_2\ \dots\ B_m] \tag{3.12}$$

Given a vector $\boldsymbol{\tau_c}$ of commanded forces and moments the thruster configuration matrix can be used to calculate the required thruster forces $\boldsymbol{u_{thr}}$ to achieve this by

$$\boldsymbol{u_{thr}} = T^{\dagger}(q)\boldsymbol{\tau_c} \tag{3.13}$$

where $T^{\dagger} = T^T(TT^T + \lambda^2 I)^{-1}$ is the damped pseudoinverse of $T(q)$. This solution corresponds to minimizing the norm of the thruster efforts and the norm of the deviation from the thrust command, with the damping part taking care of any possible problems with a singular thruster configuration matrix. For more details the reader is referred to [31].

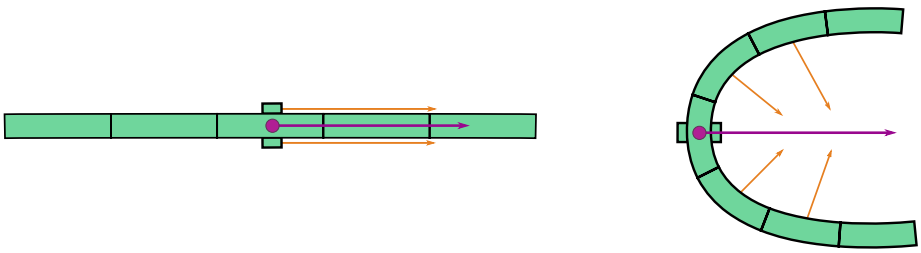


Figure 3.3: Two situations where the commanded virtual force vector (purple) is the same, but the resulting thruster actuation (orange) differ because of the USM's configuration.

Chapter 4

Path Following and Collision Avoidance

This chapter presents the main results of this thesis, namely the path following and collision avoidance controllers developed for the new and innovative underwater swimming manipulator. Both path following and collision avoidance are well-known control problems, for which there exist much research and many good solutions. However, a solution to these problems for the USM has never been presented before this thesis. The first section will present the path following controller, followed by the collision avoidance controller in the second section. Simulations demonstrating the controllers' capabilities are shown in the next chapter.

4.1 Path Following

When designing a path following control system for the USM there are two main challenges to address. Firstly, the USM is an underwater vessel, meaning that its working environment is 3D. The path following controller should thus have the ability to follow 3D paths. Secondly, in contrast to traditional underwater vehicles, the USM

has a flexible body. If possible, this flexibility should be utilized to improve the robot's path following capabilities.

The problem of 3D path following for underwater vehicles has been solved numerous times before, although mainly for vessels with single rigid bodies. The USM has the ability to continuously change its body shape by actuating its joints. However, when considering a single instant in time the body shape is constant. The USM can thus be seen as a rigid body with the ability to change its shape from one time instant to the next. With this perspective, the previously developed methods for path following can be applied to the USM as well, as long as the method is general enough to work for a vessel of arbitrary shape. The approach taken to the path following control system developed in this thesis has thus been to design a guidance system consisting of two parts: One that guides the USM along its desired path irrespective of its shape, and one that controls the shape of the USM.

The kinematic model and motion control system for the USM is influenced by the choice of base frame. Where the base frame is placed on the robot's body is important, as the configuration of the robot is controlled relative to the base frame. For instance, if the base frame is placed in the center of mass of the rearmost link on the USM, this link can be viewed as a rigid body vessel and the rest of the USM can be viewed as an attached manipulator arm that can be configured as desired. For the path following controller, the base has been placed in the COM of the center link. The links in front of and the links behind the center link are thus viewed as two separate manipulator arms, one in each direction. By achieving path following for the base frame the center of the robot will follow the desired path, and the body can be shaped symmetrically about this point.

Figure 4.1 shows the complete path following controller. The waypoint line-of-sight (WLOS) controller is the part of the system responsible for the actual path following, while the configuration controller manages the shape of the USM. It can also be observed that some alterations have been done to the original motion control framework presented in chapter 3, namely that the dynamic controller no longer receives reference velocities from the kinematic controller, but directly from the guidance system.

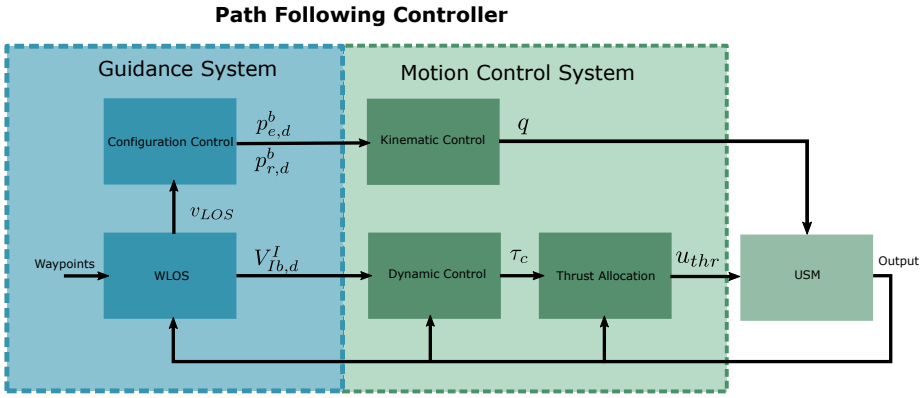


Figure 4.1: Illustration of the general structure of the path following controller. Notice that the motion control system has been lightly altered to fit with the guidance system.

4.1.1 3D Line-of-Sight Waypoint Guidance

The path following method adopted in this thesis is a modified version of the waypoint line-of-sight (WLOS) guidance algorithm. This is a well-known algorithm often used in naval guidance systems for surface vessels [12]. In this algorithm the desired path is a series of connected straight line segments, meaning that the path following controller only needs to handle straight lines. Each line segment is defined by two points in space called waypoints, and each subsequent line segment shares a common waypoint. In other words, the entire path can be defined by a series of waypoints with straight lines drawn from one waypoint to the next. The advantages of the WLOS method is that it is simple, intuitive and easy to tune. In addition, it can also be used to generate paths of more complex shapes by placing waypoints close together. The process of generating the path is called motion planning and is something that will not be addressed in this thesis, as the path will be assumed given by some higher level control entity in the control system. Figure 4.2 shows an example of a waypoint-generated path.

The WLOS guidance law supplies the motion control system with control references that guide the base frame $\mathcal{F}_b$ of the USM along a given straight-line path in 3D. The method was originally developed with vessels that are underactuated in mind, such

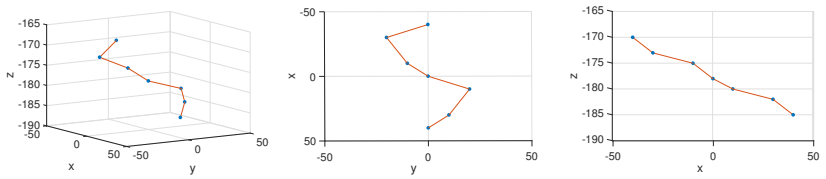


Figure 4.2: A path in 3D space defined by a set of waypoints. Left: 3-D plot. Middle: Path projected onto the xy-plane. Right: Path projected onto the xz-plane.

as surface ships, where only forward velocity and yaw rate can be controlled. In this case, the velocity vector can only be controlled by either adjusting the speed of the vessel, which affects the velocity vector's magnitude, or by adjusting the yaw angle, which influences its direction. The line-of-sight (LOS) guidance system determines a desired direction vector termed the LOS vector, that if followed will lead to path following. Based on the difference between the current velocity vector and the LOS vector the desired yaw rate can be found. There are different guidance strategies for the forward velocity, but it has been proven that the LOS guidance law will lead to path following as long as the velocity is kept larger than some nonzero value. A simple but effective guidance law for the velocity is thus to keep it at a constant reference value. The desired yaw rate and velocity are sent as reference variables to the motion control system.

In 3D the principle is the same. The LOS guidance system determines the LOS vector, and reference variables are found based on the difference between the LOS vector and the current velocity vector. Since the velocity vector is no longer in a plane but in 3D space, a representation of orientation in 3D space must be used instead of a single angle, such as a rotation matrix or unit quaternions.

In general the LOS vector can be found by using regular vector geometry. Given the position of the USM base frame origin as the vector $\boldsymbol{p}_{Ib}$, and a desired straight line path given by $\boldsymbol{l} = \boldsymbol{a} + c\boldsymbol{n}$, the shortest vector, $\boldsymbol{d}$, from the base frame to the line can be found as

$$\boldsymbol{d} = (\boldsymbol{a} - \boldsymbol{p}_{Ib}) - [(\boldsymbol{a} - \boldsymbol{p}_{Ib})^T \boldsymbol{n}]\boldsymbol{n} \tag{4.1}$$

where $\boldsymbol{a}$ is some arbitrary point on the line, $\boldsymbol{n}$ is a unit vector pointing along the line and c is a constant. Using this the LOS vector $\boldsymbol{v}$ is defined as

$$\boldsymbol{v}_{LOS} \triangleq \boldsymbol{d} + \Delta \boldsymbol{n} \tag{4.2}$$

where Δ is a control parameter called the look-ahead distance. Figure 4.3 and 4.4 shows the LOS vector in 2D and 3D space respectively.

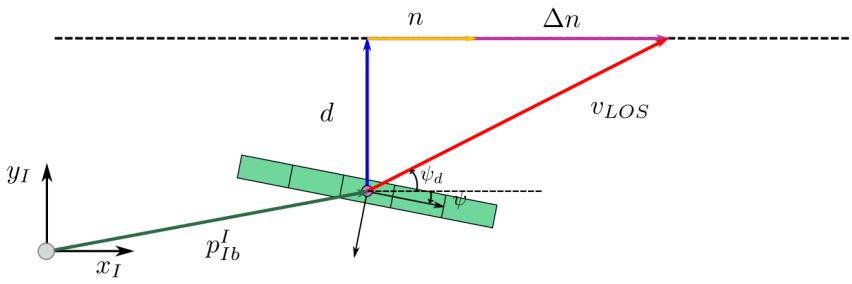


Figure 4.3: Illustration showing the LOS vector in 2D.

For the 2D case the desired yaw angle is determined by the LOS guidance law [4]

$$\psi_d = atan(\frac{d(t)}{\Delta}) \tag{4.3}$$

where the LOS vector is represented by $d(t)$ and Δ. This is illustrated in figure 4.3. From this a desired yaw rate can be found as $r_{ref} \triangleq r_r = \psi_d - \psi$, which is the reference sent to the motion control system.

In 3D the orientation of the LOS vector, and thus the desired base frame orientation, cannot be represented by a single angle. There are several ways to represent the orientation of a vector in 3D space, but in this thesis, rotation matrices are used. To represent the desired base frame orientation a new frame, $\mathcal{F}_d$, is defined, with an origin identical to the base frame and with its x-axis lying along the LOS vector. The orientation of the new frame relative to the inertial frame is represented by the rotation matrix R^I_d. Figure 4.5 (a) illustrates the desired orientation frame and the LOS vector.

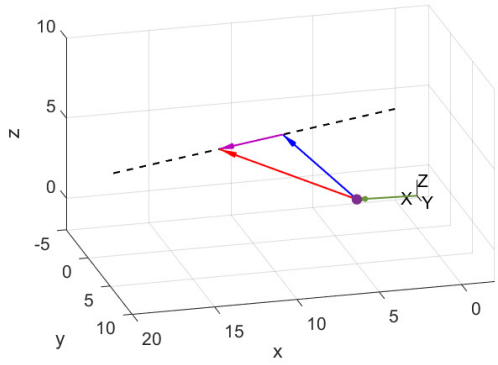


Figure 4.4: Illustration showing the LOS vector in 3D. The red vector is the LOS vector, the blue vector is $\boldsymbol{d}$, the purple vector is $\Delta\boldsymbol{n}$ and the green vector is p_{Ib}^{I}.

The desired orientation matrix R_d^I is determined through the use of passive intrinsic XZY Euler angles. By defining the elemental rotations as $R_x(\phi)$, $R_y(\theta)$, and $R_z(\psi)$, the rotation matrix can be given as

$$R_d^I = R_x(\phi)R_z(\psi)R_y(\theta) \tag{4.4}$$

The LOS vector is used to find the Euler angles. The x-axis of $\mathcal{F}_d$ lies along the LOS vector, implying that the x-axis rotation, roll, can be chosen freely. In this thesis, the roll angle is chosen such that the USM's base frame desired roll angle is zero at all times. The USM's base frame is defined with its z-axis pointing downwards when the robot is upright, which means that a roll angle of zero degrees is given by a rotation of $\phi = \pi$ around the inertial x-axis. This first elemental rotation gives a new frame with the axes x_1, y_1 and z_1. After the initial rotation around the x-axis, the yaw angle, ψ, is found by projecting the LOS vector onto the new xy-plane and finding the angle of rotation between the new x_1-axis and the projected vector. Using this angle a rotation is done around the z_1-axis. Finally, the pitch angle, θ, is found as the angle between the x_2-axis and the LOS vector. The intermediate frames used in this process are visualized in figure 4.5 (b).

The attitude deviation between the base frame and the desired frame can now be found by using the rotation matrix, and is expressed as

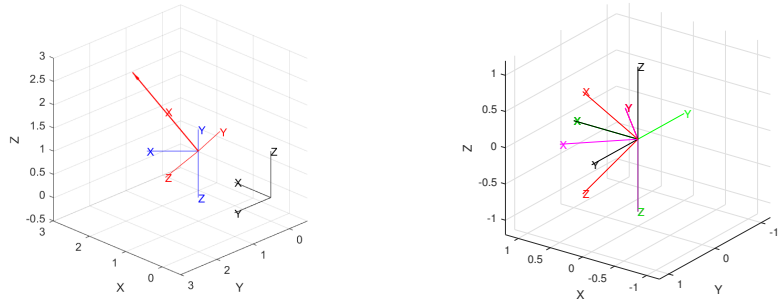
$$\tilde{R} = {R_b^I}^T R_d^I \tag{4.5}$$

The reference roll, pitch and yaw rates, p_r, q_r and r_r, are found by extracting the Euler angles from $\tilde{R}$.

A guidance law for surge speed has not been investigated in this thesis, and its reference value has thus simply been set to a constant nonzero value u_r. The output of the WLOS guidance system that is sent to the dynamic controller in the motion control system is

$$\boldsymbol{V}_{\boldsymbol{Ib,ref}}^{\boldsymbol{I}} = [u_r \; v_r \; w_r \; p_r \; q_r \; r_r]^T \tag{4.6}$$

where the heave and sway reference velocities v_r, and w_r are zero, and the other



(a) (b)

Figure 4.5: Illustration showing a LOS vector that gives Euler angles $\phi = \pi$, $\psi = -\frac{\pi}{4}$, and $\theta = \frac{pi}{4}$ for the desired base orientation. (a) Inertial frame in black, USM base frame in blue, and desired orientation frame in red. (b) Inertial frame in black, frame $x_1y_1z_1$ after the first rotation in green, $x_2y_2z_2$ after two rotations in magenta, and the desired orientation frame after all three rotations in red.

variables are determined as described above.

4.1.2 Configuration Controller

The fact that the WLOS guidance system manages to steer the USM's base frame along a desired path independent of the body's configuration means that the body shape remains free to use for other purposes. For instance, if a USM is tasked with inspecting a pipeline by following a path parallel to the pipe while using a camera mounted on the end-effector, it can simply turn the end-effector with the camera towards the pipeline without this affecting the path following. If the USM does not have any other task than path following the body configuration can be used to improve the path following performance.

For path following it is desirable to have fast path convergence and to minimize the path deviation. For autonomous systems, energy consumption is also of interest, as lowering the energy consumption leads to longer operational times. If the configuration of the USM could be used to reduce path deviation, improve path convergence and

reduce energy consumption, this would be a significant enhancement to the path following controller. Unfortunately, there are no obvious answers to how the robot configuration should be adapted to achieve these objectives.

One possible way to gain some insight into how the configuration could be used is by considering the physics of the robot system. For most marine crafts the hydrodynamic drag force is the main source of energy loss. Changing the body shape of the USM to minimize the drag force would thus most likely lead to a reduction in energy consumption. The hydrodynamic forces could also be used to more accurately control the motion of the USM in the same way that ships and airplanes use rudders and other control surfaces for steering.

The body shape's influence on hydrodynamic drag forces

The hydrodynamic drag force an object experiences when subject to a fluid flow is the sum of the drag force resulting from skin friction drag and pressure drag [6]. The skin friction drag force is a result of the fluid moving along the surface of the object, and can be compared to the regular surface friction between rigid bodies. The pressure drag force comes from the fluid hitting the surface of the object, similar to how a ball hitting a wall exerts a force on the wall. To minimize the total drag force both types of drag must be considered. However, for an object like the USM with a long and slender body, the pressure drag is dominant of the two, and should be prioritized when finding ways to reduce the drag.

Two determining factors for the total fluid drag force that an object experiences are the cross-sectional area, A_c, and angle of attack, α, of the fluid flow. The cross-sectional area represents how much fluid that hits the object's surface, while the angle of attack gives an indication of how much of the fluid that exerts pressure force and how much exerts skin friction force. Reducing the cross-sectional area reduces the amount of fluid that affects the object, thus resulting in a reduction of the total drag force. Decreasing the angle of attack leads to more skin friction drag and less pressure drag, meaning that the total fluid drag force in most cases is reduced. This is illustrated in figure 4.6.

Imagine a situation where the desired path is parallel with the body of the USM, but some distance away. This scenario can be seen in figure 4.7. Following the commands

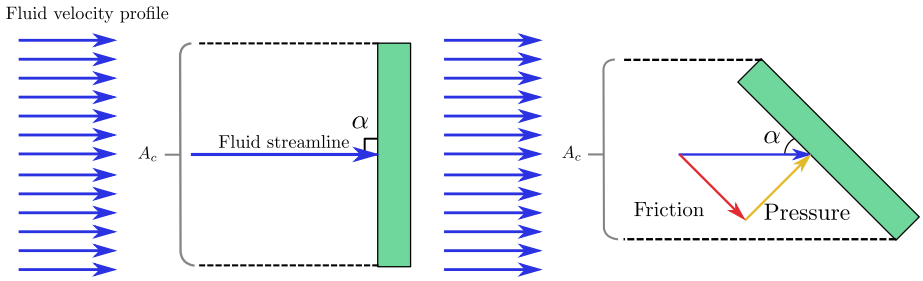


Figure 4.6: Illustration showing how the fluid flow over an object can result in different types of hydrodynamic drag forces. The blue vectors represent fluid **flow**. Left: The fluid hits the object at a ninety degrees angle of attack, and the object experiences only pressure drag force. Right: The fluid flow velocity component is decomposed to show the part of the fluid flow that contributes to frictional drag force (red vector) and which part contributes to pressure drag force (orange).

of the WLOS guidance system, the USM will start rotating its body towards the path in order to align its base frame with the LOS vector. Assume first that the USM keeps its body straight at all times. During rotation, this body shape has a large cross-sectional area and a ninety degrees angle of attack, which results in a large frictional force. One way to decrease the drag force is by curving the USM such that both ends point towards the desired path. As can be seen from figure 4.7, doing this both reduces the cross-sectional area and the angle of attack for large parts of the body. It is thus conceivable to assume that such a maneuver will reduce the total drag force experienced by the USM.

The USM's body shape can also be used to steer its trajectory. This can be done by using the front and rear part of the body in the same way as a ship uses a rudder to steer. By applying this form of steering it is not necessary to use the transversal thrusters to turn the robot. It might thus be possible to achieve more precise path following by using both the body shape and the thrusters to steer. In addition, in some cases it might be desirable to disable the use of transversal thrusters, possibly losing some accuracy in the path following, but saving power from the reduced thruster actuation. Bending the USM towards its desired path, as in figure 4.7, is how the front

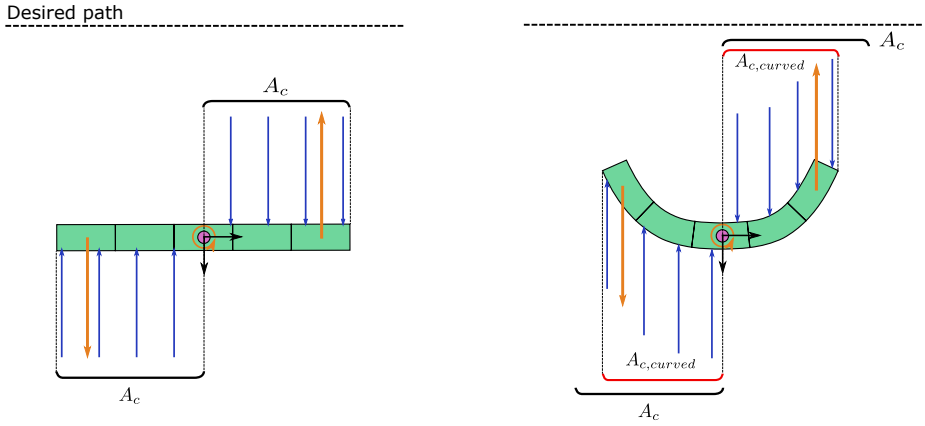


Figure 4.7: Illustration showing how the USM experience less hydrodynamic drag by curving its body. Blue vectors represent fluid flow, and the orange vectors represent thruster force/moment. Both situations show a single instant in time when the robot starts rotating. In this depiction it is assumed that all base linear velocities are zero, i.e. there is only rotational motion.

and rear part of the USM would be used as rudders to steer the robot towards the path. This curving of the USM's body thus both reduces the hydrodynamic drag force and contributes to steering the robot in the right direction, and with that shows great potential when it comes to improving the path following controller.

Configuration Guidance Law

Based on the observations made regarding the system's hydrodynamics, the configuration controller has been designed to curve the robot towards its desired path, as this is likely to reduce drag and improve the steering accuracy. This bending of the robot is achieved by using the inverse kinematics controller to solve two separate problems, one for the front part and one for the rear part of the USM. The kinematic tasks are to position the front and rear of the robot at desired reference points, $p_{f,d}$ and $p_{r,d}$, in space. These points are calculated based on the difference in orientation between the LOS vector and the x-axis of the base frame.

Let the base frame's x-axis unit vector be represented by $\boldsymbol{e_1} = [1\ 0\ 0]^T$, and let γ be the angle between the LOS vector and $\boldsymbol{e_1}$. By taking the cross product of the two vectors and normalizing the result a unit quaternion describing the rotation between $\boldsymbol{e_1}$ and the LOS vector can be found as

$$\boldsymbol{k} = cross(\boldsymbol{e_1}, \boldsymbol{v}) \tag{4.7}$$

$$\bar{\boldsymbol{k}} = \frac{\boldsymbol{k}}{||\boldsymbol{k}||_2^2} \tag{4.8}$$

$$\boldsymbol{q}_t = [\cos(\frac{c_f \gamma}{2}) \quad \sin(\frac{c_f \gamma}{2})\bar{\boldsymbol{k}}^T]^T \tag{4.9}$$

where $||\boldsymbol{k}||_2^2$ is the euclidean norm of $\boldsymbol{k}$, $\boldsymbol{q}_t \in \mathbb{R}^{4\times1}$ is a unit quaternion and c_f is a control parameter called the *curve factor*. The desired position of the front of the robot is then found by rotating a point on the base frame x-axis, described by the vector $w_f = L_f e_1$ where L_f is some constant, using the quaternion. The curve factor determines how far the vector w_f should be rotated. If $c_f = 1$ the vector will be rotated to align with the LOS vector. The desired position for the rear is found in a similar manner by rotating the vector $w_r = -L_r e_1$ with the same quaternion but in the other direction, i.e. $\boldsymbol{q}_t = [-c_f \gamma \quad \bar{k}^T]^T$.

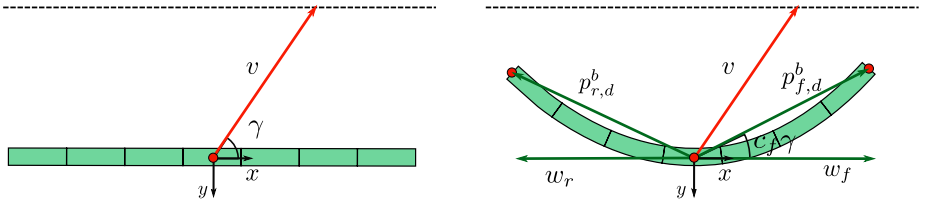


Figure 4.8: Left: The USM in a straight configuration. Right: The configuration controller makes the robot bend. The red vector is the LOS vector, the base frame is shown by the black vectors, and the green vectors are the vectors that are rotated using the quaternion.

Configuration Control

The kinematic controller is used to make the rear and front end of the USM follow their references, $p_{f,d}$ and $p_{r,d}$. As the references are position coordinates it is natural to use position as a kinematic task variable. For the front end of the USM, or what has previously been called the end-effector, such a kinematic task can be described as

$$\boldsymbol{\sigma_{pos,e}} = \eta_e = [x_e \quad y_e \quad z_e]^T \tag{4.10}$$

where x_e, y_e and z_e are the coordinates of the end-effector given in the inertial frame. To successfully implement the CLIK algorithm described in chapter 3 the task Jacobian given by the relation $\dot{\boldsymbol{\sigma}} = J_T \boldsymbol{\zeta}$ is required. The task Jacobian is easily found by making the realization that the derivative of the task variable simply is the linear velocity of the end-effector

$$\dot{\boldsymbol{\sigma}_{\boldsymbol{pos,e}}} = [u_e \quad v_e \quad w_e]^T = \boldsymbol{V_{Ie}^I}(1:3) \tag{4.11}$$

where the indexing $(1:3)$ means the first three rows of the column vector $\boldsymbol{V_{Ie}^I}$. The geometric Jacobian for the end-effector was determined in chapter 2 (eq. 2.21), and gives the relationship $\boldsymbol{V_{Ie}^I} = J_{g,e}(q)\boldsymbol{\zeta}$. By comparing this relationship with the equation above it can be seen that the task Jacobian can be expressed as

$$\dot{\boldsymbol{\sigma}_{\boldsymbol{pos,e}}} = J_T \boldsymbol{\zeta} = J_{g,e}(1:3)\boldsymbol{\zeta} \triangleq J_{pos,e}\boldsymbol{\zeta} \tag{4.12}$$

where $J_{pos,e} \in \mathbb{R}^{3\times(6+n)}$ is called the position Jacobian for the end-effector. A task and task Jacobian for positioning the rear end of the USM can be found in a similar manner, assuming that the geometric Jacobian $J_{g,r}$ describing the relationship between the rear velocities and system velocities is known.

The task Jacobians describe the relationship between the task variable and the system velocities. The system velocities also include the base velocities, meaning that any solution found by the CLIK algorithm for positioning the end-effector will include not only the required joint angles but also the reference base velocities needed to move the entire robot in a way that helps with the positioning task. In the case of the configuration controller, this is undesirable, as the WLOS guidance system takes care

of the motion of the base, while the configuration controller is intended to alter the shape of the USM's body by actuating the joints, and not influence the base velocities.

One way to handle this is simply to discard the reference base velocities found by the CLIK algorithm and only use the joint velocity references. This is however sub-optimal, as the CLIK algorithm will assume that the base can be moved as part of its solution, thus not using the joint actuation optimally. A better way to solve the problem is simply to remove the base velocities from the equation 4.12, such that the task Jacobian instead is given by

$$\dot{\boldsymbol{\sigma}}_{\boldsymbol{pos,e}} = J_T \dot{\boldsymbol{q}} = J_{pos,e}(:, 7 : (7+n))\boldsymbol{\zeta}(7 : (7+n)) \tag{4.13}$$

where $J_T \in \mathbb{R}^{3\times n}$ simply is the the last n columns of $J_{pos,e}$ and $\dot{\boldsymbol{q}}$ is the joint velocities. The CLIK algorithm will then try to solve the positioning problem using the joints only, most likely using the joints in a more optimal way.

With the two task Jacobians, one for the positioning of the end-effector and one for the positioning of the rear, the kinematics controller solves the two separate inverse kinematics problems using the CLIK algorithm 3.5 presented in chapter 3

$$\dot{\boldsymbol{q}}_{\boldsymbol{ref,e}} = \boldsymbol{J_{T,e}}^{\dagger}(\dot{\sigma_{e,d}} - K_{IK,e}\dot{\tilde{\sigma}_e}) \tag{4.14}$$

$$\dot{\boldsymbol{q}}_{\boldsymbol{ref,r}} = \boldsymbol{J_{T,r}}^{\dagger}(\dot{\sigma_{r,d}} - K_{IK,r}\dot{\tilde{\sigma}_r}) \tag{4.15}$$

where $\dot{\boldsymbol{q}}_{\boldsymbol{ref,e}}$ contains the reference joint velocities for the joints between the base and the end-effector and $\dot{\boldsymbol{q}}_{\boldsymbol{ref,r}}$ contains the reference joint velocities for the joints between the base and the rear of the USM. The joint velocities are integrated to get joint position references, which are sent to the joint controller. In this thesis the joint controller is assumed to be an internal part of the mechanism in each joint, and thus not discussed. This assumption also coincides with the workings of the simulation software that has been used. The result is simply that all the joint angles follow their reference values with a very small time delay.

4.1.3 Control Parameters

The path following control system has a total of six control parameters that can be tuned to get the desired behavior. The control parameters are found in both the guidance system and in the motion control framework. Both the structure of the control system and the system dynamics leads to coupling effects between some of the parameters. This fact makes the system difficult to tune, and little effort has been put into finding optimal values for the control parameters in this thesis.

The WLOS guidance system contains a single control parameter, the look-ahead distance Δ. The effect of this parameter can best be understood by looking at figure 4.3. By having a small Δ the angle between the LOS vector and the vector $\boldsymbol{d}$ will be small, resulting in the USM making aggressive turns towards the path. This can lead to fast path convergence, but also increases the risk of overshooting the path. Decreasing the look-ahead distance can be compared to increasing the proportional gain of a PID controller.

The configuration controller is tuned by adjusting the curve factor c_f. With a curve factor of $c_f = 1$ the USM tries to align the end-effector with the LOS vector, and with a $c_f = 0$ the USM will keep its body straight. Because the curve factor influences how much the front and rear part of the USM is moved relative to the base, it indirectly influences the motion of the base frame. For instance, rotating the front part towards the desired path will generate a reactive hydrodynamic force that rotates the base in the opposite direction. The changes in body configuration should thus not be too large, as they will affect the WLOS controller's ability to successfully do path following. Through simulations and tuning by trial and error, it has been found that the curve factor should not exceed $cf = 0.5$.

In the kinematic controller, there are the two inverse kinematic gain matrices, $K_{IK,e}$ and $K_{IK,r}$. These gain matrices work in a similar way to the proportional gain in a PID controller. Having a large gain will result in fast joint actuation, but will also influence the base more because of the kinematic coupling between the base and the rest of the body.

The dynamic controller can be compared to a PD controller and contains the usual

proportional and derivative gains K_P and K_D. They are most easily chosen as diagonal matrices, with one diagonal entry for each of the six base frame velocity variables. The dynamic controller generates force references, and the resulting thruster actuation can thus be controlled by scaling the gains in the dynamic controller.

4.2 Collision Avoidance

Collision avoidance (CA) is an integral part of any mobile autonomous system, as collisions can lead to damage to both robot and environment. One major challenge for collision avoidance systems is their dependence on the environment. An effective CA system for a robot that works in an environment with large single-body obstacles will certainly be very different than that for a robot whose job is to find the center of a maze without hitting the walls. If a robot works in an environment with very different types of obstacles the CA system must either be flexible enough to handle the different situations, or it must use different solutions based on the situation.

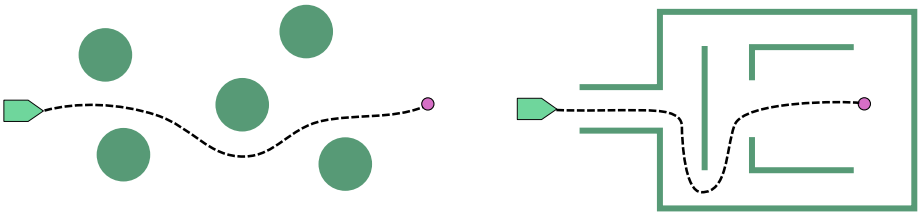


Figure 4.9: Illustration of two types of environments with obstacles of very different geometric nature.

The USM will need to handle collision avoidance in very different types of environments. As for most naval vessels, it will spend much time traversing open waters, which are areas where there are few other objects relative to the size of the area. Typical examples of these sort of situations are like when the USM moves from its docking station to a subsea oil structure, or when it is doing inspection work following a pipeline. Most of the time this means avoiding a few objects in an otherwise open space.

In addition, the USM needs to be capable of collision avoidance in cluttered and confined spaces, like when it is traversing the structural steel cage of a subsea gas compressor, moving within a cave or exploring the insides of a sunken ship. In this case, there are many obstacles to consider simultaneously, while the USM has less room to maneuver in, making it a difficult problem to solve. These type of environments are perhaps especially interesting, considering the fact that one of the USM's main advantages is its flexibility body and its resulting ability to move in tightly enclosed areas that would otherwise be unavailable.

This thesis presents a reactive collision avoidance system that is mainly designed for maneuvering in confined spaces and to take advantage of the USM's flexibility. Although designed for cluttered environments, the CA system is general enough to handle any type of obstacles and environments.

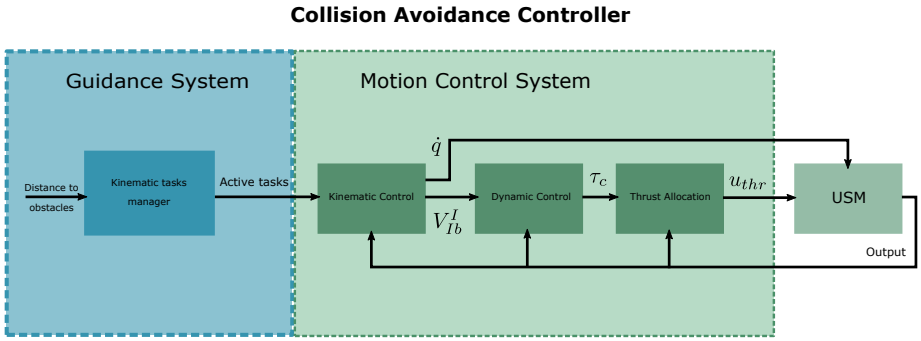


Figure 4.10: Illustration of the general structure of the collision avoidance controller.

4.2.1 Inverse Kinematics Collision Avoidance Controller

It has previously been mentioned that the USM can be viewed as a floating base manipulator arm, and the method of using inverse kinematics to control these types of redundant robotic systems has been discussed in chapter 3, as well as earlier in this chapter in regards to the configuration controller. The ability to define multiple kinematic tasks and solve them in a prioritized order to exploit the robot's redundancy

makes the inverse kinematics an ideal method for implementing collision avoidance.

Being based on inverse kinematics control, the collision avoidance system works by defining kinematic tasks that when solved ensure both collision avoidance and motion towards a goal point. Kinematic tasks are defined for points. Assuming that all the kinematic tasks are satisfied, collision avoidance is still only guaranteed for the points where they are defined. Adding more points will, in theory, ensure CA for a larger part of the robot, but at the same time, this will decrease the redundancy of each point, reducing the probability of the system being able to satisfy all the kinematic tasks. To implement CA for the entire robot, kinematic tasks are thus defined for points strategically placed along the USM's body, called *collision avoidance points* (CAPs). For instance, considering a USM with five modules three CAPs could be placed, one at the end-effector, one at the USM center of mass and one at the end of its tail.

Kinematic Tasks for Collision Avoidance

The type of kinematic task used to implement collision avoidance in this thesis is the position norm task. This task is simply to control the norm between a point on the USM, one of the defined CAPs, and a point $\boldsymbol{p_o}$ in space, which can be described as

$$\boldsymbol{\sigma_n} = \sqrt{(\boldsymbol{p_o} - \boldsymbol{p}_{CAP})^T(\boldsymbol{p_o} - \boldsymbol{p}_{CAP})} \in \mathbb{R}^1 \tag{4.16}$$

where $\boldsymbol{p}_{CAP}$ and $\boldsymbol{p_o}$ is the position coordinates of the CAP and the point respectively, given in the inertial frame. To determine the task Jacobian the time derivative of the task variable is taken and found to be

$$\dot{\boldsymbol{\sigma}}_{\boldsymbol{n}} = \frac{(\boldsymbol{p_o} - \boldsymbol{p}_{CAP})^T}{||\boldsymbol{p_o} - \boldsymbol{p}_{CAP}||_2^2}\dot{\boldsymbol{p}_{CAP}} \tag{4.17}$$

The derivative of the CAP position can be recognized as the derivative of a positioning task, and by using equation 4.11 and 4.12 it is seen that $\dot{\boldsymbol{p}_{CAP}} = \dot{\boldsymbol{\sigma}_{pos}} = J_{pos}\boldsymbol{\zeta}$, which implies that

$$\dot{\boldsymbol{\sigma}}_{\boldsymbol{n}} = \frac{(\boldsymbol{p_o} - \boldsymbol{p}_{CAP})^T}{||\boldsymbol{p_o} - \boldsymbol{p}_{CAP}||_2^2}J_{pos}\boldsymbol{\zeta} = J_n\boldsymbol{\zeta} \tag{4.18}$$

The position norm task can be used to achieve much of the behavior desired in a collision avoidance system, as it enables the creation of both attractive and repulsive action. By defining a task where the desired position norm is zero, the resulting inverse kinematics solution will move the CAP towards the point $\boldsymbol{p_o}$, trying to reach that exact position. If instead, the desired norm is nonzero, the IKs will try to keep the given distance from the point $\boldsymbol{p_o}$. This means that as long as the norm is less than the desired norm, the IK will push the CAP away from $\boldsymbol{p_o}$. Figure 4.11 illustrates how the position norm task with a nonzero desired norm affects the system in three different situations.

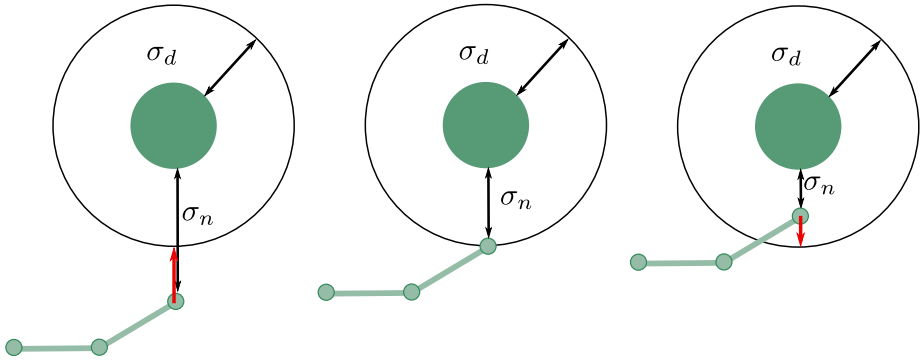


Figure 4.11: Illustration showing how a position norm task σ_n with a nonzero desired norm σ_d affects the system in three different cases. The green line represents the USM, and the three dots along it the CAPs. The black lines are distance norms, and the red vector shows in which direction the IK solution will try to move the CAP.

The Collision Avoidance Algorithm

The collision avoidance algorithm uses CAPs and their defined position norm tasks to achieve collision avoidance. For each CAP i a collision avoidance task $\tau_{CA,i}$ is defined, which is a position norm task with a nonzero desired position norm. The norm for these tasks is the norm between the CAP and the closest obstacle surface point. In addition, at least one directional task $\tau_{D,i}$ is defined for one of the CAPs, which is a position norm task with a desired norm equal to zero. The directional task is used to

maneuver the USM in the right direction so that the robot manages to not only avoid a collision but also to pass the obstacle(s).

Since a CA task will pull its CAP towards the closest obstacle if the CAP is further away from the obstacle than the desired minimum distance, as can be seen in figure 4.11, it is important to use the task only when needed. Let $\mathcal{T}$ be the set of all the position norm tasks that at some point during operation could be needed. At each time step the collision avoidance algorithm determines which of the tasks in $\mathcal{T}$ that are needed, and in which priority they should be solved. These tasks are called *active tasks*, and are defined by the set $\mathcal{A} \subseteq \mathcal{T}$.

The tasks in the active set are sent to the inverse kinematics controller. As with the path following controller the inverse kinematics problem must be split into a front part and rear part since the base frame has been chosen to be in the center link. Active tasks for CAPs from the base and forward are solved as one problem for the front part of the USM, while the rest of the tasks, which belong to CAPs behind the base, are solved as another. Both problems are solved according to the SRMTP method (eq. 3.5), but the base reference velocities are only used from the solution to the first problem. This means that for the rear IK problem only the joint velocities need to be considered, as in equation 4.14.

To make sure that the USM moves towards its goal the directional task is always active and part of the active tasks set. The collision avoidance tasks, however, are activated only when a collision is plausible and deactivated when it is safe to say that a collision will not happen.

The CA tasks are activated if their norm becomes smaller than the desired value, meaning that the CAP is closer to the obstacle than desired. This can be expressed as

$$\tau_{CA,i}(t) \in \mathcal{A} \quad if \quad \sigma_{CA,i}(t) < R_{o,i} \tag{4.19}$$

where $R_{o,i}$ is a chosen minimum desired distance to an obstacle for CAP i. When a CA task is activated it is desirable to keep it active until it is safe to say that the obstacle has been passed. This is achieved by comparing the direction to the obstacle with the direction to the desired goal position. When the minimum desired distance to the

obstacle is achieved, and the obstacle is no longer in the way of the desired travel trajectory, the CA task is deactivated. This can be expressed as

$$\tau_{CA,i}(t) \notin \mathcal{A} \quad if \quad \sigma_{CA,i}(t) >= R_{o,i} \quad and \quad \boldsymbol{v_{n,i}}^T \boldsymbol{v_{d,i}} >= 0 \tag{4.20}$$

where $\boldsymbol{v_{n,i}}$ is the vector between the CAP and the closest obstacle surface point, and $\boldsymbol{v_{d,i}}$ is the vector between the CAP and the desired position.

Figure 4.12 depicts how the collision avoidance tasks are activated and deactivated according to the rules presented in equations 4.19 and 4.20. The figure illustrates a scenario where a directional task is defined for CAP 1 with the red point as the desired position, and one collision avoidance task is defined for each of the three CAPs. The task set is thus $\mathcal{T} = \{\tau_{D,1}\ \tau_{CA,1}\ \tau_{CA,2}\ \tau_{CA,3}\}$. At the first time instant none of the CA tasks are activated, as all the CAPs are further away from the obstacle than the distance R_o, and the active set is simply $\mathcal{A}_{t_1} = \{\tau_{D,1}\}$. In the second time instant, the first CAP has come closer to the obstacle than desired, and the CA task has been activated, trying to push the CAP directly outwards from the obstacle to a distance R_o as shown by the red vector. At the same time, the directional task tries to move the CAP towards the desired position. The active task set is now $\mathcal{A}_{t_2} = \{\tau_{CA,1}\ \tau_{D,1}\}$. By the SRMTP method, where the CA task is given top priority, the CAP is allowed to move towards the goal as long as the distance to the obstacle is increasing towards R_o, resulting in the circular motion. In the third time instant CAP 1 has reached the desired distance from the obstacle but is still active as the obstacle is still in the way of the CA. In other words, if the CAP were to move directly towards the goal, it would start decreasing the distance to the obstacle again. Furthermore it can be seen that the CA task of the second CAP has also been activated, and the active task set is $\mathcal{A}_{t_3} = \{\tau_{CA,1}\ \tau_{CA,2}\ \tau_{D,1}\}$. In the final time instant the CA task of CAP 1 is deactivated, as the obstacle has been successfully passed according to the defined rules 4.20, and $\mathcal{A}_{t_4} = \{\tau_{CA,2}\ \tau_{D,1}\}$.

4.2.2 Control Parameters

The control parameters for the collision avoidance control system are the inverse kinematic gains for the kinematic tasks and the minimum desired distance to an

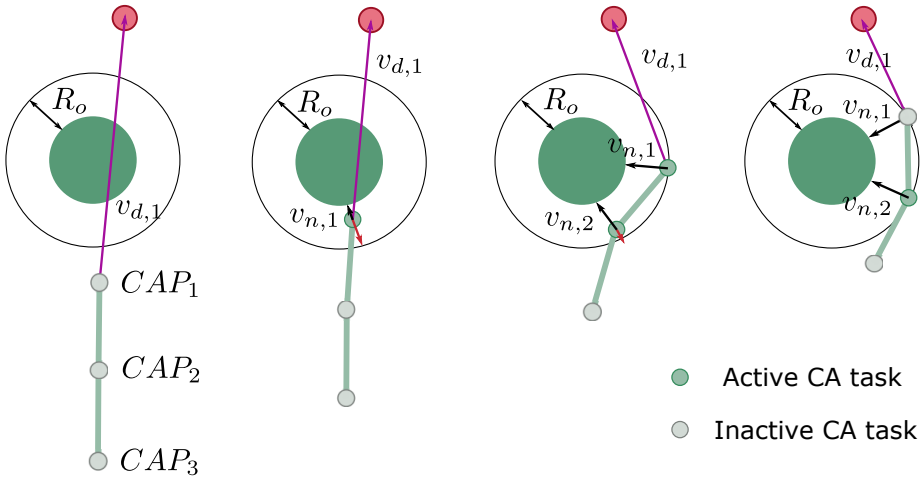


Figure 4.12: Illustration that shows when collision avoidance tasks are active.

obstacle. The control parameters greatly influence each other. By increasing the IK gains larger reference velocities will be generated by the IK controller, and assuming that the dynamic controller is able to track the references, this means that the USM can do fast maneuvers. When the USM can react quickly the safety margins can be reduced, or in other words, the minimum desired obstacle distance can be reduced. However, these fast maneuvers lead to high thruster actuation, which is undesirable with respect to energy consumption and material fatigue. With smaller IK gains a lower energy consumption is achieved, but the minimum distance to obstacles should be increased in order to be sure that there won't be any collision.

In addition to the coupling between the control parameters themselves, their choice should also be influenced by the velocities at which the USM moves. If the USM is moving with a high velocity it is likely that large IK gains or a wide safety radius are needed to avoid a collision. Likewise, if the USM is moving slowly it might be sufficient with a very small obstacle distance or low IK gains.

Because of the coupling between the control parameters and their dependence on the velocity of the robot it would be advantageous with a control mechanism that

adapts the control parameters based on the situation. If the USM is in open waters moving at a high speed the minimum desired distance to an obstacle can be made large, so that the robot has time to adjust for any obstacle without having to actuate the thrusters too much. If the USM is about to enter a cluttered area with many obstacles the velocity can be lowered to allow for maneuvers closer to the obstacles. In addition, the IK gains might be increased, enabling the USM to move very close to the obstacles and utilize the available space. Methods for such adaptive parameter control has not been investigated and is left for future research.

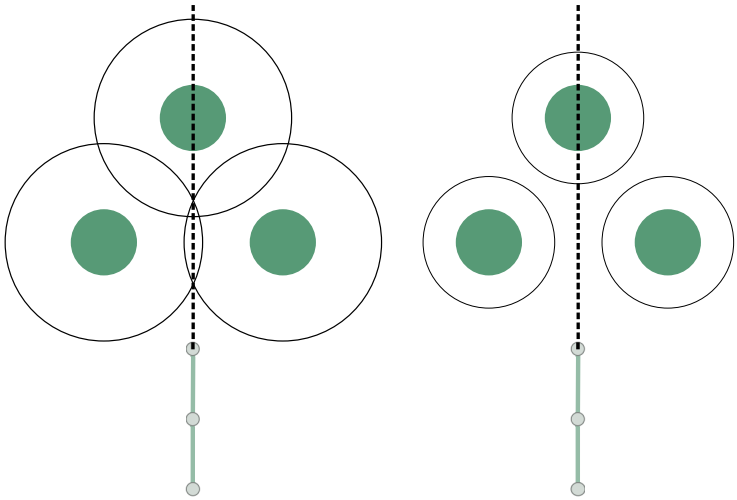


Figure 4.13: Illustration showing how the desired minimum obstacle distance can influence the behavior of the collision avoidance system. In the left scenario, the USM will try to avoid the left and right obstacle, although they are not really in the way. In the right scenario, the USM will only try to avoid the front obstacle, which is in the path and should correctly be avoided. However, if the velocity of the USM is large and the IK gains are small it might not apply enough power to steer away before a collision occurs, because of the short reaction distance.

Chapter 5

Implementation and Simulations

Simulations have been used to test and validate the path following and collision avoidance controllers developed in this thesis. The USM and its dynamics have been modeled and simulated in the simulation software Vortex, while the control system has been implemented using Matlab/Simulink. This chapter will first briefly introduce the simulation setup before several simulation cases demonstrating the performance of the controllers are presented.

5.1 Simulation Model: Vortex

In order to obtain accurate simulation results when testing the guidance and control system, it is essential to have a good model of the system. The underwater swimming manipulator and its many degrees of freedom make it difficult to model, especially considering the complex hydrodynamics that arises for such a mechanism. Analytical models for both the USM [32] and its relative, the underwater snake robot [18], has been developed for control purposes, but are so far limited to motion in the plane. A

different approach to modeling the USM is to use modeling and simulation software and is the approach taken in this thesis. The simulation software that has been used to model the USM and its dynamics is called Vortex.

Vortex is a software created by CM labs [8] that provides a 3D environment with realistic physics suitable for both modeling and simulating various mechanisms. In addition to providing a dynamics engine for land-based mechanisms, Vortex also supports marine simulations through the use of computational fluid dynamics (CFD) to simulate hydrodynamic forces. The hydrodynamic forces that are calculated by Vortex are the nonlinear drag force and added mass effect. The software runs simulations in real time, which limits the complexity of the CFDs employed because of the computational restraints. However, the detailed geometry of the robot is taken into consideration when determining the hydrodynamic force, and its accuracy is thus still quite good. In [15] it was shown that, by proper tuning of fluid parameters, it is possible to create a Vortex model of a USR that is accurate enough to give simulation results that are comparatively similar to those collected from experiments using the real world robot. Considering this it is not unreasonable to conclude that a model of the USM in Vortex is a decent representation of the robot to be used for simulation purposes. An image of the Vortex model of the USM can be seen in figure 5.1.

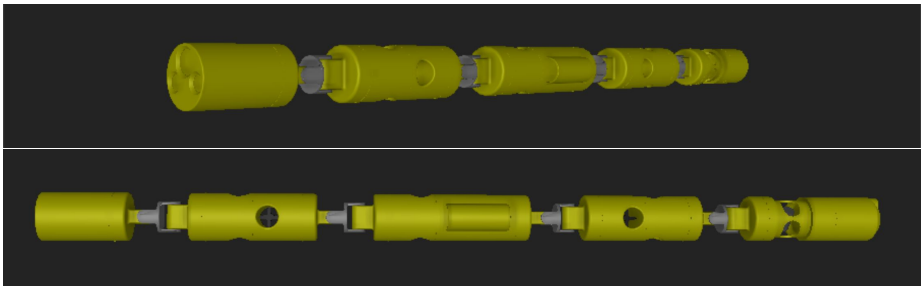


Figure 5.1: The USM model in Vortex.

5.2 Control System: MATLAB/Simulink

MATLAB/Simulink has been used to design the control system used in simulations. Simulink is a well known graphical programming environment by MathWorks for modeling, simulating and analysing dynamical systems. Simulink is tightly linked with the programming language MATLAB, which can be used to script Simulink.

The reason for using MATLAB/Simulink is two-fold. Firstly, the program is familiar to the author and widely used in the automatic control community. Secondly, the Vortex software facilitates direct two-way communication with Simulink, making it the easiest tool to integrate with Vortex. The Vortex model can read control inputs from the Simulink model and send simulation outputs to the Simulink model in real time. Since the simulation state data is sent to Simulink it can be processed, stored and analysed directly using Matlab.

All control systems rely on measurements of different state variables. Using a simulation environment like Vortex means that all system states are available. The states that are needed in the control systems developed in this thesis are the joint angles, base position and orientation and base velocities, and are thus sent from Vortex to Simulink. Based on this, position, orientation and velocity of other points on the USM can be calculated by using the kinematic model, but for simplicity, this information has also been extracted directly from Vortex for the end-effector and rear of the USM.

5.3 Simulation Study: Path Following

One of the main goals of this thesis has been to develop a path following controller for the USM, and the resulting system was presented in the previous chapter. In this section, the performance of the path following controller will be investigated by applying the controller to different path following scenarios.

5.3.1 Evaluation Criteria

As the goal of this section is to analyse the performance of the path following controller, it is important to establish the criteria on which the performance will be evaluated.

With the main objective of the controller being path following, it is, of course, essential that the USM manages to follow the path without large deviations. An important metric for the path following controller's performance is thus *path deviation*.

A second performance metric that is almost equally as important is the *energy consumption*. Energy consumption directly influences the robot's operational time and is very important for the cost efficiency of the robot. The USM's main energy consumption comes from thruster actuation. When doing path following it is thus desirable for the USM to have as low thruster actuation as possible, while at the same time managing to accurately follow the desired path.

To achieve both minimal path deviation and energy consumption is not possible, as the two optimization objectives are adverse. Perfect path following requires fast and precise movements, which can only be achieved through extensive thruster actuation. However, the right use of the USM's body shape does help improve the performance with respect to both criteria, as will be shown by the simulations in this section.

5.3.2 Scenario: Waypoint Navigation

The first scenario was created specifically to test the capabilities and limitations of the path following control system. Waypoints were thus chosen to create a path that would test all the qualities of the controller:

- The path is 3D.
- The path has long straight segments to test the controller's ability to converge to a steady state.
- The path has wide turns which are the most likely encountered in a real-world operation.
- The path has narrow turns, including a change of direction, to test the controller's peak performance.

The control system is modular, consisting of the 3D line-of-sight waypoint guidance controller (WLOS) and the configuration controller. The WLOS controller is designed to

be able to achieve path following regardless of the configuration of the USM, assuming that the configuration maintains a required number of degrees of freedom. To highlight this simulations have been run both with and without the configuration controller active.

The control parameters that have been used are identical for all path following simulations and can be seen in table 5.1. The inverse kinematics gain K_{IK} has been used for both kinematic tasks, i.e. positioning the front and rear of the USM.

Table 5.1: Control parameters used for path following simulations.

Parameter	Description	Value
Δ	Look-ahead distance	3
c_f	Curve factor	0.3
K_{IK}	Inverse kinematics gain	$diag([0.5\ 0.5\ 0.5])$
K_P	Proportional gain	$diag([0.5\ 0.5\ 0.5\ 0.13\ 0.12\ 0.12])$
K_D	Derivative gain	$diag([0\ 0\ 0\ 0.13\ 0.13\ 0.13])$

Case: Disabled Configuration Controller

In the first case that will be presented the configuration controller was disabled, and the USM's configuration set to be straight, i.e. all the joint angles are zero throughout the entire simulation. In this setting, the USM resembles a submarine shaped AUV, and can only use its thrusters to navigate.

The simulation result can be seen in figure 5.2, where the red and green points are the base frame starting and end point, the blue points are the waypoints, the orange lines are the desired path and the blue line is the actual base trajectory. The waypoints are used to define the desired path and direction of movement along the path, so the controller will not try to directly reach the waypoints. Because of this, the first waypoint is never reached. Additionally, the last waypoint is not reached simply because the simulation was terminated before the USM had traveled far enough along the path.

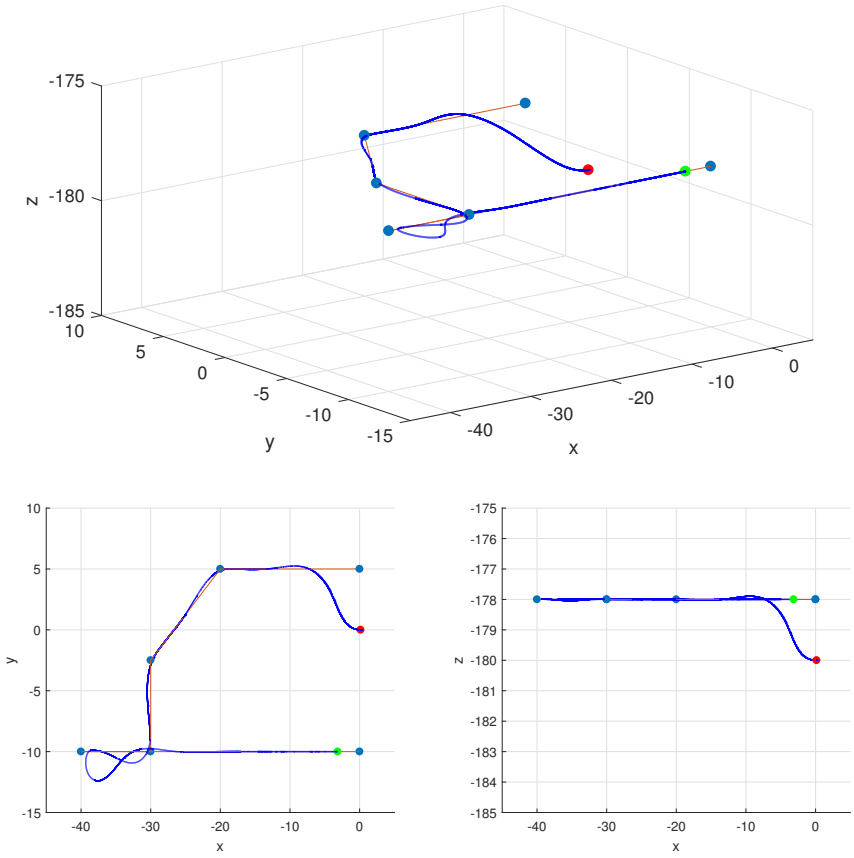
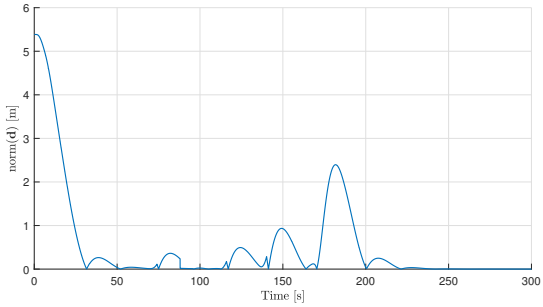


Figure 5.2: Path following with the configuration controller disabled. The red and green points are the base frame starting and end point, the blue points are the waypoints, the orange lines are the desired path and the blue line is the actual base trajectory.

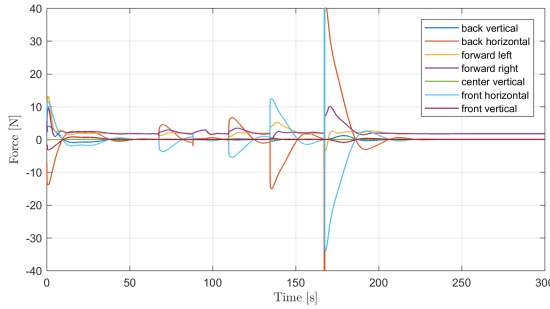
It is obvious from figure 5.2 that the USM manages to converge to and follow the path, which shows that the proposed WLOS controller works as intended. To get a better sense of the controller's performance the path deviation and thruster actuation can be studied. Figure 5.3 contains plots of the two evaluation criteria, with (a) showing the path deviation, (b) the individual thruster actuation and (c) the total thruster actuation, which is the sum of each individual thruster actuation.

The path deviation in (a) is best understood in context with the trajectory shown in figure 5.2. The USM starts some distance away from the desired path, which can easily be recognized as a large path deviation in 5.3 (a). As the USM moves towards the desired path the deviation decreases, reaches zero and then increases again. This first increase in path deviation is the overshoot that can be seen in 5.2 as the USM first hits the path. After the overshoot, the path deviation decreases towards zero until another large increase in deviation occurs. This pattern repeats itself, where the second increase is the overshoot after the first turn, the third and fourth increase is because of the following two turns, and the last and largest increase comes from the 180 degrees turn. After the last turn, the USM manages to converge to a path deviation of less than $1cm$, which can be regarded as almost perfect path following when considering the fact that the radius of the robot's body is approximately $8.5cm$. NB: Before each turn overshoot there is a small "spike" in the path deviation. This is when the USM transitions from one path segment to the next, and is a result of the path deviation being calculated as the shortest distance to the path.

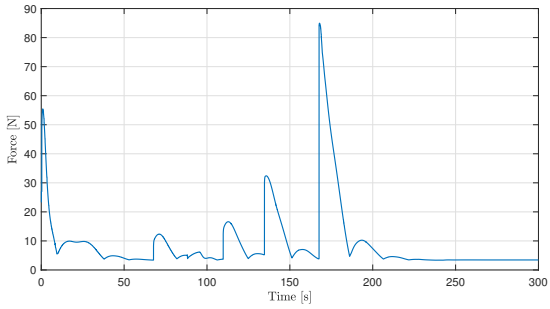
Similarly, looking at the thruster actuation in figure 5.3 (b) and (c) the same phases of the trajectory can easily be identified. During turns, the thruster actuation increases proportionally with how sharp the turn is, and at the end of the simulation when the USM has converged to the path the thruster actuation stays at a low even level to keep the forward velocity. Looking at the individual thruster actuation in (b) it is also easy to see the that the thruster allocation system tries to optimize the thruster actuation. During the turns, which happens in a xy-plane, the front, and back horizontal thrusters are mainly used. This is the optimal approach, as these thrusters will generate the largest torque forces around the base origin.



(a) Path deviation calculated as the smallest distance to the path at each time instant.



(b) The actuation of the seven different thrusters. There is a saturation limit at ±40 Newton.



(c) The total thruster actuation of all thrusters added together.

Figure 5.3: Path deviation and thruster actuation for the trajectory shown in figure 5.2.

Case: Active Configuration Controller

One of the USM's advantages to more traditional AUV and ROVs is its highly maneuverable and flexible body. This agility can be used to access narrow and hard to reach areas, but it is also proposed in this thesis that the body shape can be used to improve the robot's path following capabilities. The configuration controller is an attempt at accomplishing this. Its effect on the USM's path following performance is investigated by running a simulation with the same path as before, but this time with the configuration controller active, and comparing the results with the first simulation.

Figure 5.4 shows the two trajectories for the base origin, with the configuration controller disabled in blue and with the configuration controller active in red. Looking closely at the respective trajectories it seems like that activating the configuration controller might yield performance improvements. The effect of the configuration controller can better be seen by inspecting the path deviation and total thruster actuation shown in figure 5.5. From the path deviation plot in (a) it can easily be seen that using the configuration controller improves the path following accuracy. Drawing conclusions from the thruster actuation plot in (b) is harder, as the two graphs are very similar in nature. The magnitude of the thruster actuation seems to be identical in both cases, but looking closer it can be observed that the spikes in thruster actuation are shorter in time with the configuration controller active.

The total accumulative path deviation and thruster actuation can be found by calculating the area under the curves in figure 5.5. The resulting numbers can be seen in table 5.2. From these values, it can be calculated that by using the configuration controller there is approximately a 20 percent decrease in path deviation and a 15 percent decrease in thruster actuation. This is a substantial amount, especially considering thruster actuation and the resulting energy conservation. This shows that actively using the flexibility of the USM during path following can lead to significant performance improvements.

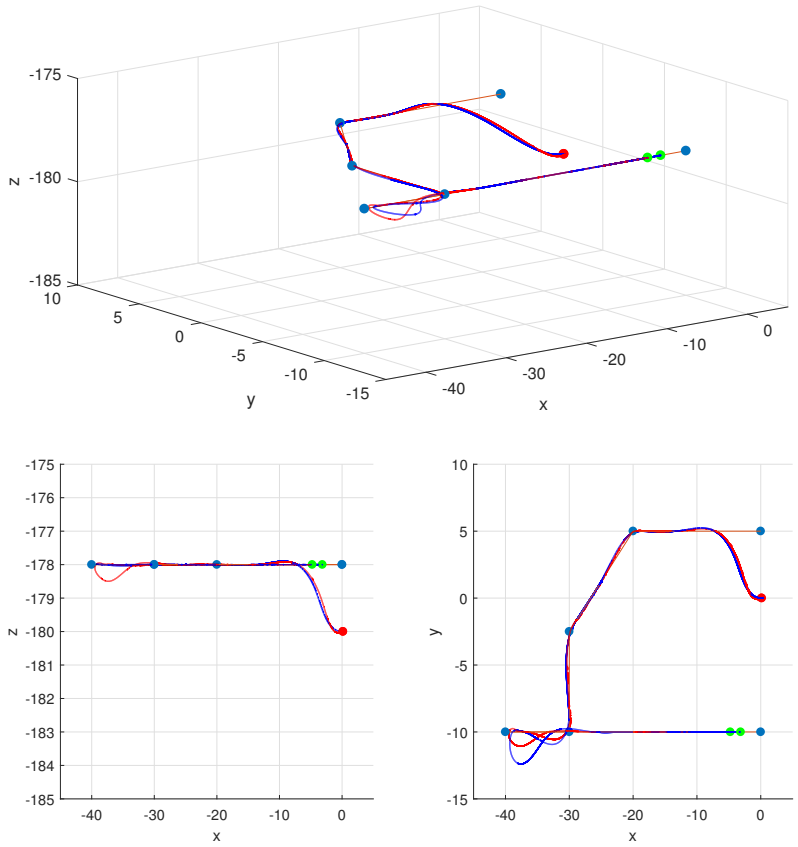
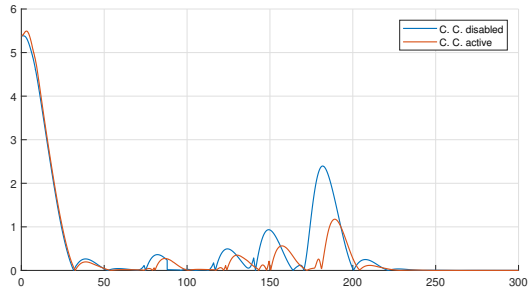


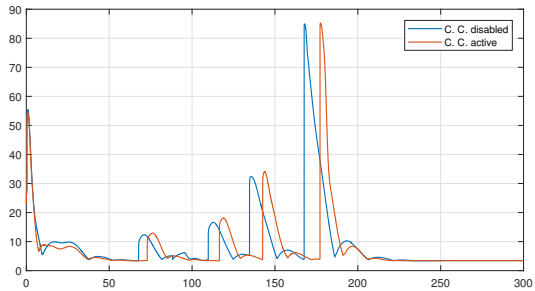
Figure 5.4: Desired path and actual trajectory of the base origin viewed in 3D, xy-plane and xz-plane respectively.

Table 5.2: The accumulated path deviation and thruster actuation, with the configuration controller (CC) disabled and active.

Performance Metric	CC disabled	CC active	active/disabled [%]
Path deviation	162.3	130.6	80.5
Thruster actuation	2811.7	2403.9	85.5



(a) Path deviation calculated as the smallest distance to the path at each time instant.



(b) The total thruster actuation of all thrusters added together.

Figure 5.5: Performance metrics.

5.3.3 Scenario: Spiral Path Docking

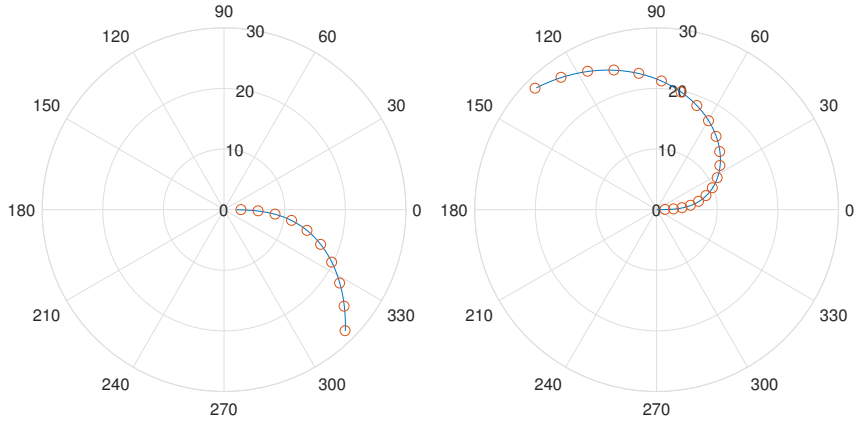
In the first simulation scenario, the path following controller was tested on a path constructed specifically to assess its capabilities. In this scenario, the applicability of the controller will be tested on a path inspired by a possible real-world task for the USM. One of the ideas associated with the autonomous USM for use in the subsea oil industry is to build a docking station for the robot on the seafloor. This docking station will be used to recharge the USM and serve as a base of operations. In order to dock, the robot must navigate into a tube on the station.

In the paper *Spiral path planning for docking of underactuated vehicles with limited FOV* [28] a method for performing the docking procedure is presented. The method proposed finds a spiral path to follow into the docking station, that always fulfills two important properties: The docking station is always within the field of view (FOV) of the onboard sensor, and there is no curvature on the path when it reaches the docking station. This path planning method has been used to generate spiral paths that the path following controller will be tested on.

Case: Planar Spiral Path

The path planning method in [28] was used to generate two different planar spiral paths. The path following controller developed in this thesis can only follow paths consisting of straight line segments defined by a set of waypoints. The spiral paths thus had to be converted to waypoints, which was done simply by sampling points along the spiral paths. The points were sampled evenly spaced with respect to the radius from the docking station to the spiral path. Figure 5.6 shows the two spiral paths and the generated waypoints, where in both cases the docking station is located in the origin with its opening pointing along the positive x-axis.

The simulation results can be seen in figure 5.7. In (a) the initial conditions of the USM are: base position aligned with the first waypoint at $(x, y) = (20, -20)$, body straight and parallel with the y-axis, with the end-effector pointing in the positive y-direction. It can be seen that the base of the USM follows the spiral path closely without problems.



(a) Spiral path 1.

(b) Spiral path 2.

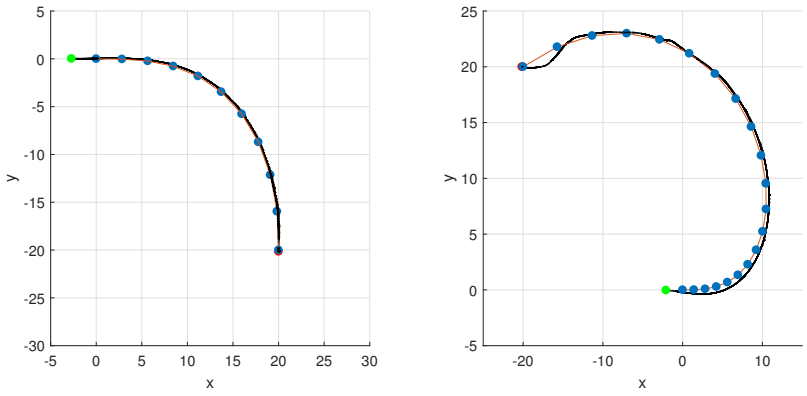
Figure 5.6: Planned spiral paths with generated waypoints superimposed.

For the second spiral path, with results shown in figure 5.7 (b), the USM's base starting position is at $(x, y) = (-20, 20)$, with its front pointing in the direction parallel with the positive x-axis. Also for this case, the USM's base follows the spiral path quite closely. Notable deviation can only be seen along the last part of the spiral, where the curvature of the spiral is at its largest.

These results are obtained without any alterations to the path following controller, showing that even though the controller is designed for following simple straight line segments, these can be connected together to create more complex paths that can be followed without large path deviations.

Case: 3D Spiral Path

As the path following controller is a full 3D controller it should be able to follow 3D spiral paths as well. Similarly, as before, a planar spiral path is found by the path planning algorithm found in [28], and waypoints are generated along the spiral. The 3D spiral is then created by offsetting the waypoints in the z-direction. This distribution in the z-direction is done uniformly in a chosen interval defined by the z-coordinate of the



(a) (b)

Figure 5.7: Desired path, waypoints and actual base trajectory results.

starting position of the USM and the z-coordinate of the docking station. This process does not preserve the properties of the planar spiral path, but this is not of interest in this case, as the extension to 3D is simply to test the path following controller's 3D capabilities following a complex path.

The simulation results can be seen in figure 5.8, where the path is a 3D extension of the spiral path used previously, shown in figure 5.2 (b). The starting position of the USM's base is again $(x, y) = (-20, 20)$ with a starting depth of $z = -170$ and its front pointing in a direction parallel to the xy-plane in the positive x-direction. The position and depth of the docking station is $(x, y, z) = (0, 0, -180)$. Examining the figure it can be observed that the path following controller guides the base of the USM along the path without large path deviations, also in 3D.

5.4 Simulation Study: Collision Avoidance

To develop a collision avoidance control system for the USM has been the second main goal of this thesis, and the resulting collision avoidance algorithm was presented in chapter 4.2. Similarly, as for the path following system, the collision avoidance

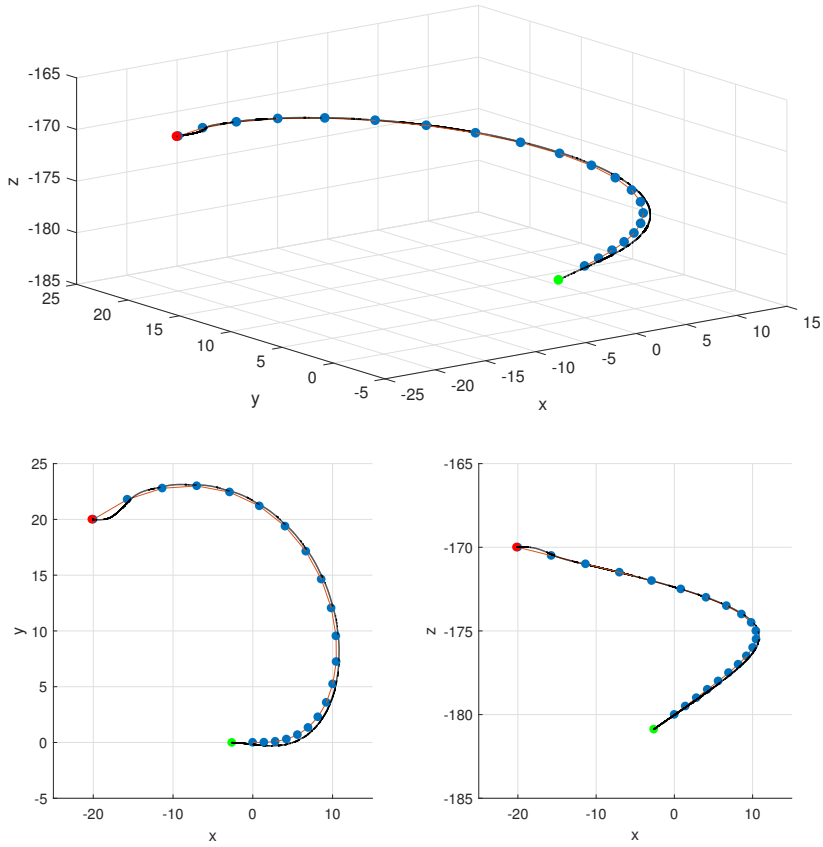


Figure 5.8: Desired path and actual trajectory of the base origin viewed in 3D, xy-plane and xz-plane respectively.

algorithm will be validated through simulations.

In all simulations in this section, it is purely the collision avoidance system that is tested. In other words, the collision avoidance controller is assumed to be active at all times, and the desired position of the robot is assumed given by some higher level control entity. Furthermore, three collision avoidance points (CAPs) have been defined for the USM, one at the end-effector, one in the COM of the center link and one at the rear of the tail, illustrated by the red dots in figure 5.9. These CAPs will be given the subscripts e, b and r respectively when referenced. For instance, when talking about the collision avoidance task belonging the end-effector CAP the notation used is $\tau_{CA,e}$.

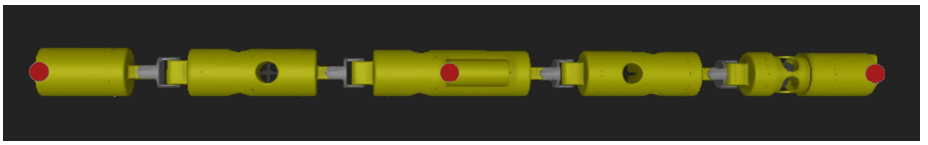


Figure 5.9: Illustration showing the defined collision avoidance points (CAPs) for the USM during simulations.

5.4.1 Scenario: Single Obstacle

To best demonstrate how the collision avoidance controller works, and how the control parameters influence its behavior, simulations have been done with a single obstacle of a simple shape. Three different cases will be used to highlight the system's properties. In the first case, the effect of adding collision avoidance tasks to additional CAPs is shown, the influence of the control parameters is more closely inspected in the second case, and in the third and final case, the system's ability to handle arbitrarily shaped obstacles is demonstrated.

Case: Adding Collision Avoidance Tasks

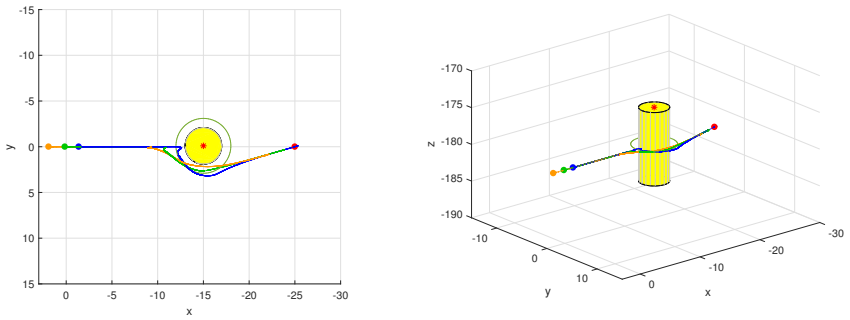
In the first case, two simulations are run in identical environments, where the only difference between them is the number of CA tasks that have been defined for the

USM. For the first simulation, only a single CA task has been defined for the end-effector CAP. In the second simulation, a CA task has been defined for each of the three CAPs. In addition, for both simulations a directional task is defined for the end-effector, to create the forward motion. The two task sets are thus $\mathcal{T}_a = \{\tau_{CA,e}\ \tau_{D,e}\}$ and $\mathcal{T}_b = \{\tau_{CA,e}\ \tau_{CA,b}\ \tau_{CA,r}\ \tau_{D,e}\}$ respectively.

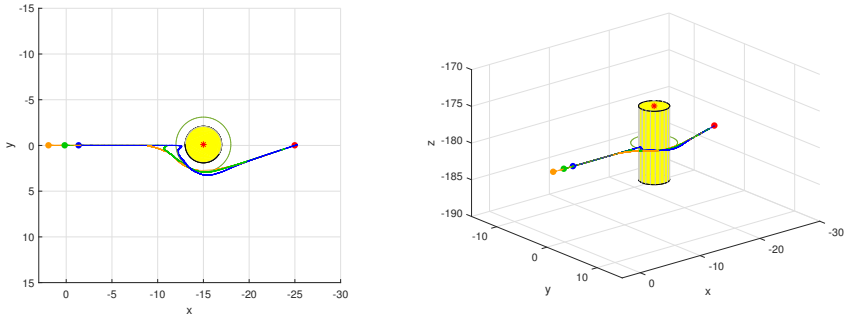
The two simulations are shown in figure 5.10, where (a) shows the simulation with task set $\mathcal{T}_a$ and (b) with $\mathcal{T}_b$. The starting point of the three CAPs on the USM can be seen as the blue, green and orange dot for the end-effector, base, and rear respectively, and the lines of the same color shows their trajectories. The minimum desired distance to the obstacle for all CA tasks is $R_o = 1$, and is visually represented as the green circle around the cylinder. The inverse kinematic gains are set to one for all tasks, and the desired position of the end-effector is shown as the red dot.

In both simulations it can be seen that the USM moves in a straight line towards the goal position until the end-effector comes closer to the obstacle than desired, i.e $\sigma_{CA,e} < R_o$. When this happens the CA task for the end-effector is activated and included in the inverse kinematics solution with the highest priority. The effect can easily be seen, as the end-effector clearly stops its movement towards the obstacle, and then starts moving away from it towards the desired distance. Since the directional task is active as well reference velocities are generated that move the end-effector towards the goal position but don't contradict the CA task, resulting in the circular movement around the obstacle.

There are no collisions in either of the two simulations, but in (a) this is purely coincidental, and it can be observed that the rear is much closer to the obstacle than desired when it passes. The effect of adding collision avoidance tasks for the base and rear CAPs can clearly be seen by comparing (a) and (b). In (b) the CA tasks are activated when the CAPs come too close and keep the desired distance from the obstacle. In addition, it should be observed that this has no large influence on the path of the end-effector, which is a result of the robot's high level of redundancy.



(a) Two tasks defined: One CA task and one directional task for the end-effector (blue).



(b) Four tasks defined: One CA for each of the three CAPs and one directional task for the end-effector (blue).

Figure 5.10: Collision avoidance with a single obstacle. $R_o = 1$, $K_{CA} = 1$, $K_{dir} = 1$

Case: Effect of the Control Parameters

In this case two identical simulations has again been run, but now with different inverse kinematic gains. The setup is similar to the previous case, with $\mathcal{T} = \{\tau_{CA,e}\tau_{CA,b}\tau_{CA,r}\tau_{D,e}\}$, $R_o = 2$ and $K_{dir} = 1$. The two simulations can be seen in figure 5.11, where the left part of the figure shows the first simulation with $K_{CA} = 0.1$ and the right part of the figure shows the second one with $K_{CA} = 2$. To avoid cluttering the plot only the trajectory of the end-effector CAP is shown.

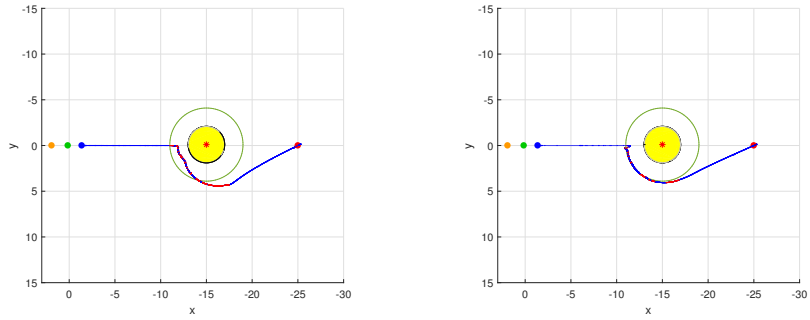
The influence of the inverse kinematics gains can clearly be seen by inspecting figure 5.11. In both simulations, the USM moves with the same velocity towards the obstacle. As expected a high gain leads to a faster response when the CA task is activated, and there is a small overshoot of the minimum desired distance before it is corrected. The closest distance to the obstacle is 0.99 and 1.54 meters for the low and high gain cases respectively. By having a larger CA task gain the desired minimum distance to obstacles can be reduced while maintaining the same level of assurance that no collision will occur. With the high gain, the minimum desired distance could be set to $R_o = 0.5$, and a collision would still not happen for this specific scenario.

Case: Arbitrary Obstacle Shape

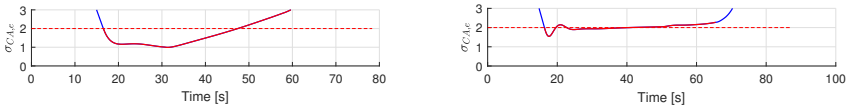
In the preceding simulations, a cylinder-shaped obstacle has been used, and the movement of the robot has been in the plane, resulting in the interpretation of the obstacle as a circle. The collision avoidance algorithm does however not put any requirements on the shape of obstacles, all it needs to function is the distance to the surface of the obstacle.

To demonstrate this a simulation similar to the previous ones has been conducted, where the upright cylinder has been replaced by a randomly shaped obstacle that has been created by connecting two box shapes and one cylinder shape. The parameters for the simulation are $\mathcal{T} = \{\tau_{CA,e}\ \tau_{CA,b}\ \tau_{CA,r}\ \tau_{D,e}\}$, $R_o = 1$, $K_{dir} = 1$ and $K_{IK,i} = 1$.

The results can be seen in figure 5.12, where (a) shows the obstacle and trajectories in 2D and 3D, while (b) shows the distance from the CAPs to the obstacle. The USM avoids the obstacle and reaches its goal position, while all the CAPs keep their



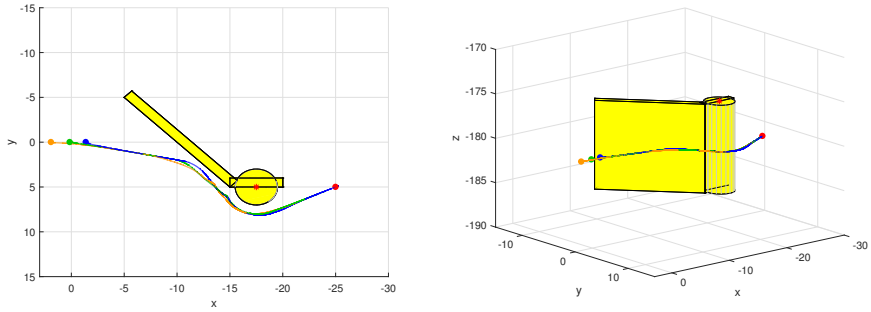
(a) Trajectory of the end-effector CAP. In the left plot $K_{CA,e} = 0.1$ and in the right plot $K_{CA,e} = 2$,



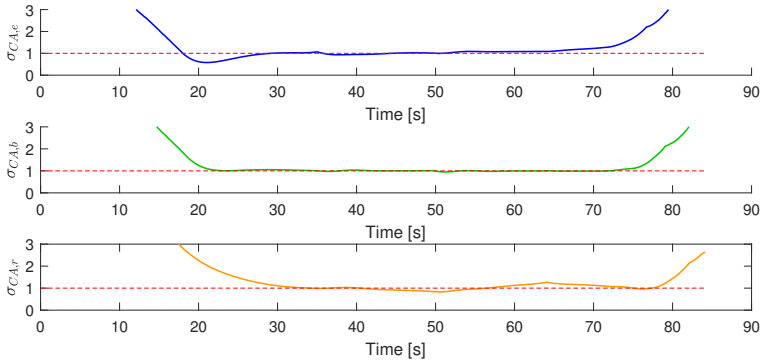
(b) Distance from the end-effector CAP to the obstacle. The dotted line is the minimum desired distance.

Figure 5.11: Collision avoidance with a single obstacle. The red part of the line illustrates when the CA task is active. Parameters: $R_o = 2$, $K_{dir} = 1$.

minimum desired distance to the obstacle.



(a) Bird's eye view and a 3D view of the obstacle, goal position and CAP trajectories.



(b) Distance from the CAPs to the obstacle. The dotted line is the minimum desired distance.

Figure 5.12: Collision avoidance with a single randomly shaped obstacle. Parameters: $R_{o,i} = 1$, $K_{dir,e} = 1$.

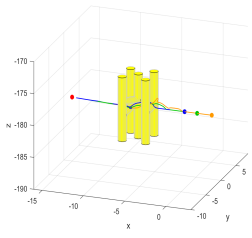
5.4.2 Scenario: Multiple Obstacles

One of the main design goals for the collision avoidance system developed in this thesis has been to make it functional in tight and enclosed areas. This entails using the flexibility and maneuverability of the USM, enabling it to enter spaces that would

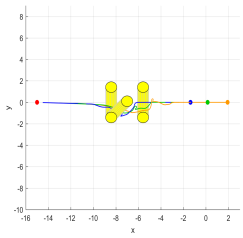
not otherwise be possible. The system is to a large extent able to achieve this, which will be demonstrated through simulations where the USM moves through an obstacle course so narrow that it must bend its body to get through.

The obstacle course was created by connecting several cylinder-shaped objects together in a small area of space. Figure 5.13 shows a simulation with the obstacle course. Looking at figure (b) it can be seen that the structure consists of five vertical cylinders and three horizontal ones. Each cylinder has a diameter of one meter. The vertical cylinders are placed close together in a radius of one meter from the central one. The three horizontal cylinders are placed at different altitudes, but all within a range of two meters on the z-axis. The task set for the simulation is $\mathcal{T} = \{\tau_{CA,e}\, \tau_{CA,b}\, \tau_{CA,r}\, \tau_{D,e}\}$, the desired minimum distance from any obstacle is set to $R_o = 0.5$ for all three CAPs and the gains are $K_{CA,i} = 2$ and $K_{dir,e} = 1$.

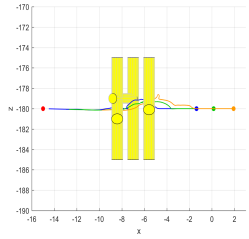
The USM approaches the structure in a straight line when end-effector CAP encounters a horizontal cylinder. It moves upward in order to pass above it, but soon encounters a vertical cylinder. The USM then moves to the left to navigate around the newly encountered vertical cylinder. However, upon performing that maneuver a second horizontal cylinder is met, resulting in the USM having to move downward to go underneath it. Lastly a maneuver to the right is need to avoid a second vertical cylinder, and the USM stays above the third horizontal cylinder, until at last it can move freely towards its goal position. The distances to the closest obstacle can be seen for each CAP throughout the simulation in figure 5.13 (g). At no point in time does a collision occur for any of the CAPs, although there are times when some of the CAPs are fairly close.



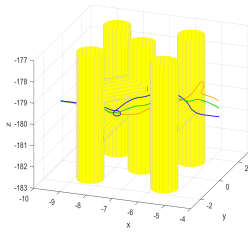
(a) 3D view.



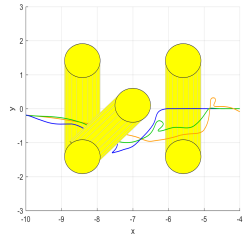
(b) Top view.



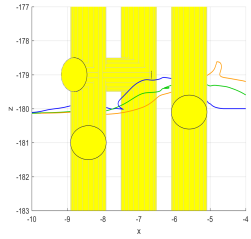
(c) Side view.



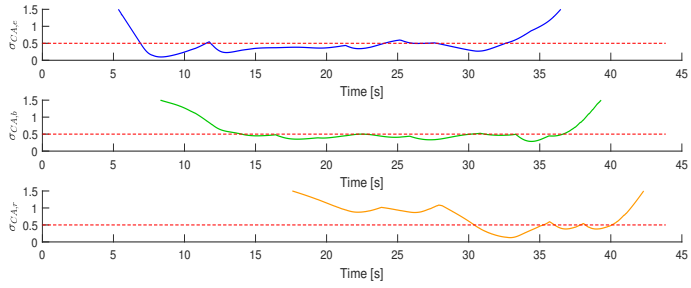
(d) Closeup: 3D view.



(e) Closeup: Top view.

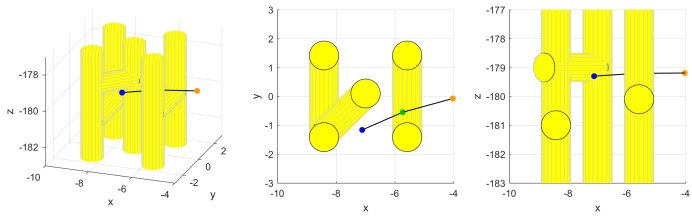


(f) Closeup: Side view.

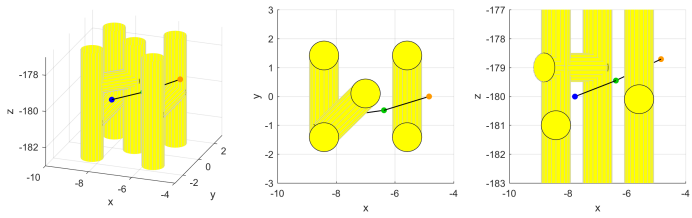


(g) CAP distances to obstacle.

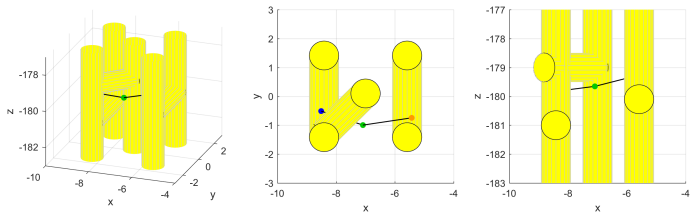
Figure 5.13: Collision avoidance in a tight and cluttered area with multiple obstacles. Parameters: $R_{o,i} = 0.5$, $K_{CA,i} = 2$, $K_{dir,e} = 1$.



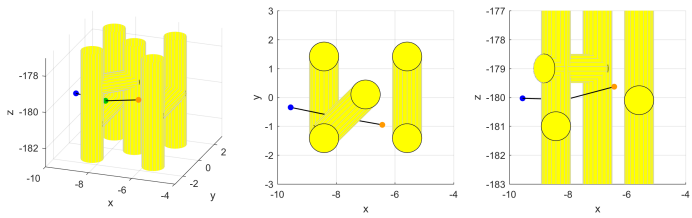
(a) t = 19



(b) t = 28



(c) t = 32



(d) t = 35

Figure 5.14: The figure shows snapshots of the simulation from figure 5.13. A simplification of the USM has been drawn as the CAPs with lines between them. Each instant in time is shown from three different viewpoints, identical with the viewpoints in 5.13 (d), (e) and (f).

Chapter 6

Conclusion

This thesis has presented path following and collision avoidance controllers for the novel USM robot. The control systems were presented in chapter 4 and validated by simulations in chapter 5. This chapter will summarize the main results of this thesis, and discuss some thoughts on future extensions and improvements to the systems.

6.1 Results

Path following for autonomous underwater vehicles is a problem that has been solved many times for a number of different vessel designs. The main difference between earlier underwater vehicles and the USM is that the USM has a flexible body, giving it the ability to change its shape. However, at a given instant in time the shape of the USM is constant, and the robot can be considered as a rigid body in the same way as most other underwater vehicles. In addition, being a highly actuated system the USM enjoy 6-DOF movement in almost all possible body shapes.

The approach taken to designing the path following controller in this thesis was thus to view the USM as a rigid body with the ability to change its shape. Path following was solved using methods for rigid bodies, while the novel shape-shifting ability was utilized to optimize the path following with respect to some desired properties. A 3D

line-of-sight controller was created for the path following, while a controller termed the configuration controller was created to manage the shape of the USM. When path following is the robot's sole objective it makes sense to optimize the task by minimizing path deviation and energy consumption. The configuration controller was designed to achieve this by shaping the USM's body to reduce hydrodynamic drag.

The simulation results in section 5.3 shows that the path following controller tracks paths with good accuracy, both simple straight line paths with turns and more complex paths like the spiral. The effect of the configuration controller was investigated by comparing simulations with the configuration controller activated in one case and deactivated in another. It was found that by using the configuration controller both thruster usage and path deviation decreased, implying that the controller does improve the USM's path following capabilities. For the path and control parameters used in the comparing simulations, the decrease was approximately 15% in thruster actuation and almost 20% in path deviation, which is a significant improvement. The numbers can, of course, be less prominent for other paths or control parameters, but the large improvement does serve as a testimony to the configuration controller's effect, and to the usage of the USM's body shape to optimize task execution in general.

The collision avoidance controller developed in this thesis has been designed with the novel properties of the USM in mind, namely the flexibility of the body and the many degrees of freedom. To this end, the algorithm has been designed for environments where these properties are most needed, in enclosed spaces with many obstacles and little room for movement. The important observation was made that the USM can be viewed as a floating manipulator arm, and as such well-known inverse kinematics control methods from the field of land-based manipulator arms were applicable. These methods are suitable for control when the USM is moving in cluttered environments, as they take advantage of the robot's available redundancy and enables solving complex tasks by considering multiple simple tasks simultaneously.

The collision avoidance controller presented in this thesis uses inverse kinematics to control points on the USM's body called collision avoidance points. The CAPs are chosen strategically, in the hope that by assuring collision avoidance of all the CAPs, no part of the robot will collide. A number of inverse kinematic tasks are defined for

the CAPs, and the collision avoidance controller works by activating the tasks when needed. The SRMTP scheme is used to solve the active IK tasks in a prioritized order.

Several simulations have been run to test the proposed CA system, both with a single obstacle and with multiple obstacles creating a tight and cluttered environment. The simulations show that the CA system successfully avoids collisions for the CAPs, even in narrow and confined spaces. This shows the potential of the method, and also demonstrates the ability of the USM to reach locations inaccessible for other vehicle designs.

6.2 Future Work

For the USM to be fully autonomous control systems are needed for a wide range of different tasks. This thesis has proposed a solution for path following and collision avoidance, which are essential building blocks for any autonomous mobile robotic system. Natural extensions to this system are motion planning algorithms that generate paths, algorithms for processing sensor data from cameras or sonars, localization and mapping, controllers for specific tasks like pipe monitoring and many more. However, the path following and collision avoidance controllers do have their limitations, and some suggestions for improvements to these specifically will now be suggested.

The biggest drawback with the path following controller is that it is not designed to handle any form of external disturbances like ocean currents. One way to remedy this is by extending the 3D line-of-sight guidance law with integral effect, enabling it to deal with constant irrotational ocean currents [5]. In the presence of currents, it is likely that the USM's configuration should be used differently to optimize for the desired properties, and the configuration controller would also need to be extended to handle this situation.

The collision avoidance controller works well but does not have any form of higher level guidance. To avoid collisions all CA tasks are given priority above the positioning task in the SRMTP scheme. This means that situations can arise where there is no solution available through the null space of the collision avoidance tasks that bring the USM closer to its desired position, effectively locking the robot in space and resulting

in it never reaching its goal. An algorithm for handling these situations should be developed to make the system more robust.

Several improvements can be made to the inverse kinematics solver. For instance, a set-based SRMTP method is proposed in [2], where it is argued that including all the tasks in the inverse kinematics solution does not always yield the optimal solution, and an algorithm for finding the optimal solution that still does not break any constraints is proposed. Another possible improvement could be a mechanism to better prioritize CA tasks, as it in certain situations might be advantageous to for instance prioritize CA of the base CAP over the end-effector CAP. As mentioned in the discussion of control parameters for the CA system, having a mechanism that actively adapts the parameters can also potentially lead to great performance improvements.

References

[1] Antonelli, G. [2014]. *Underwater Robots*, Vol. 96, third edit edn, Springer.

[2] Antonelli, G., Moe, S. and Pettersen, K. Y. [2015]. Incorporating set-based control within the singularity-robust multiple task-priority inverse kinematics, *2015 23rd Mediterranean Conference on Control and Automation (MED)* pp. 1132–1137.

[3] Børhaug, E., Pavlov, A. and Pettersen, K. Y. [2008]. Integral LOS control for path following of underactuated marine surface vessels in the presence of constant ocean currents, *Proceedings of the IEEE Conference on Decision and Control* pp. 4984–4991.

[4] Breivik, M. and Fossen, T. I. [2008]. Guidance laws for planar motion control, *Proceedings of the IEEE Conference on Decision and Control* pp. 570–577.

[5] Caharija, W., Pettersen, K. Y., Bibuli, M., Calado, P., Zereik, E., Braga, J., Gravdahl, J. T., Sørensen, A. J., Milovanović, M. and Bruzzone, G. [2016]. Integral line-of-sight guidance and control of underactuated marine vehicles: Theory, simulations and experiments, *IEEE Transactions on Control Systems Technology* **24**(5): 1623–1642.

[6] Cengel, Y. A. and Cimbala, J. M. [2006]. *Fluid Mechanics: Fundamentals and Applications*, 1st edn, McGraw-Hill.

[7] Chiaverini, S. [1997]. Singularity-robust task-priority redundancy resolution for

real-time kinematic control of robot manipulators, *IEEE Transactions on Robotics and Automation* **13**(3): 398–410.

[8] *CM Labs Webpage* [2017].
URL: *https://www.cm-labs.com/*

[9] Colome, A. and Torras, C. [2015]. Closed-loop inverse kinematics for redundant robots: Comparative assessment and two enhancements, *IEEE/ASME Transactions on Mechatronics* **20**(2): 944–955.

[10] De Angulo, V. R. and Torras, C. [1997]. Self-calibration of a space robot, *IEEE Transactions on Neural Networks* **8**(4): 951–963.

[11] Fiorini, P. and Shiller, Z. [1998]. Motion planning in dynamic environments using velocity obstacles, *International Journal of Robotics Research* **17**(7): 760–772.

[12] Fossen, T. I. [2002]. *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*, Marine Cybernetics, Trondheim.

[13] Fox, D., Burgard, W. and Thrun, S. [1997]. The dynamic window approach to collision avoidance, *IEEE Robotics Automation Magazine* **4**(March): 23–33.

[14] From, P. J., Gravdahl, J. T. and Pettersen, K. Y. [2014]. *Vehicle-Manipulator Systems: Modeling for Simulation, Analysis, and Control*, Springer.

[15] Hoffmann, B. H. [2017]. Modelling of the Underwater Snake Robot Mamba with and without Added Effectors.

[16] Johansen, T. A., Perez, T. and Cristofaro, A. [2016]. Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment, *IEEE Transactions on Intelligent Transportation Systems* **17**(12): 3407–3422.

[17] Kelasidi, E., Liljebäck, P., Pettersen, K. Y. and Gravdahl, J. T. [2017]. Integral Line-of-sight Guidance for Path Following Control of Underwater Snake Robots: Theory and Experiments, *IEEE Transactions on Robotics* **33**(3): 610–628.

[18] Kelasidi, E., Pettersen, K. Y., Gravdahl, J. T., Strømsøyen, S. and Sørensen, A. J. [2017]. Modeling and Propulsion Methods of Underwater Snake Robots, *2017 IEEE Conference on Control Technology and Applications (CCTA)* .

[19] LaValle, S. M. [1998]. Rapidly-Exploring Random Trees: A New Tool for Path Planning.

[20] Maciejewski, A. a. and Klein, C. a. [1985]. Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments, *The International Journal of Robotics Research* **4**(3): 109–117.

[21] Moe, S., Caharija, W., Pettersen, K. Y. and Schjolberg, I. [2014]. Path Following of Underactuated Marine Underwater Vehicles in the Presence of Unknown Ocean Currents, *Proc. 33rd International Conference on Offshore Mechanics and Arctic Engineering* **7**(2): 1–10.

[22] Moe, S. and Pettersen, K. Y. [2016]. Set-Based Line-of-Sight (LOS) Path Following with Collision Avoidance for Underactuated Unmanned Surface Vessel, *2016 24th Mediterranean Conference on Control and Automation (MED)* pp. 402–409.

[23] Nakamura, Y. and Hanafusa, H. [1987]. Task-Priority Based Redundancy Control of Robot Manipulators, *The International Journal of Robotics Research* **6**(2): 3–15.

[24] Orin, D. E. and Schrader, W. W. [1984]. Efficient Conputation of the Jacobian for Robot Manipulators, *International Journal of Robotics Research (IJRR)* **3**(4): 66–75.

[25] Peter E. Hart, Nils J. Nilsson and Bertram Raphael [1968]. Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions on Systems Science and Cybernetics* **4**(2): 100–107.

[26] Porta, J., Ros, L. and Thomas, F. [2005]. Inverse kinematics by distance matrix completion, *Proceedings of CK2005, 12th International Workshop on Computational Kinematics* pp. 1–9.

[27] Rao, R. S., Asaithambi, A. and Agrawal, S. K. [1998]. Inverse Kinematic Solution of Robot Manipulators Using Interval Analysis, *Journal of Mechanical Design* **120**(1): 147.

[28] Sans-Muntadas, A., Kelasidi, E., Pettersen, K. Y. and Brekke, E. [2017]. Spiral path planning for docking of underactuated vehicles with limited FOV, *2017 IEEE Conference on Control Technology and Applications (CCTA)* (August): 732–739.

[29] Sanz, P. [2009]. *Robotics: Modeling, Planning, and Control*, Vol. 16, Springer.

[30] Siciliano, B. and Slotine, J.-J. E. [1991]. The General Framework for Managing Multiple Tasks in High Redundant Robotic Systems, *Fifth International Conference on Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR* pp. 1211 – 1216 vol.2.

[31] Sverdrup-Thygeson, J., Kelasidi, E., Pettersen, K. Y. and Gravdahl, J. T. [2016a]. A control framework for biologically inspired underwater swimming manipulators equipped with thrusters, *IFAC-PapersOnLine* **49**(23): 89–96.

[32] Sverdrup-Thygeson, J., Kelasidi, E., Pettersen, K. Y. and Gravdahl, J. T. [2016b]. Modeling of underwater swimming manipulators, *IFAC-PapersOnLine* **49**(23): 81–88.

[33] Sverdrup-Thygeson, J., Kelasidi, E., Pettersen, K. Y. and Gravdahl, J. T. [2017]. The Underwater Swimming Manipulator: A Bioinspired Solution for Subsea Operations, *IEEE Journal of Oceanic Engineering* pp. 1–16.

[34] Ulbrich, S., De Angulot, V. R., Asfour, T., Torras, C. and Dillmann, R. [2009]. Rapid learning of humanoid body schemas with kinematic bézier maps, *9th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS09* pp. 431–438.