NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

TTK4550 - ENGINEERING CYBERNETICS, SPECIALIZATION PROJECT

Autonomous navigation and mapping for underwater snake robots

Marcus Engebretsen

supervised by
Dr. Kristin Y. Pettersen

co-supervised by
Dr. Pål Liljebäck

Bottom of the page 17. juni 2019

Abstract

The underwater swimming manipulator (USM) is an emerging technology that is under rapid development towards subsea applications. It is a versatile robot that can be applied to multiple purposes such as inspection and maintenance. Fundamental tasks to perform these operations are; planning a feasible and collision free path, and following it. This project investigates fundamental criteria for the path planner and consequently how to interface the path planner with a higher level control system. In particular, how to specify criteria for feasible curved paths are proposed along with a framework for path following of these paths. The goal is to provide a framework with practical utility that can be implemented on an actual USM. Kinematic simulations are finally provided to validate the correctness of the design.

Innhold

Tabeller

Figurer

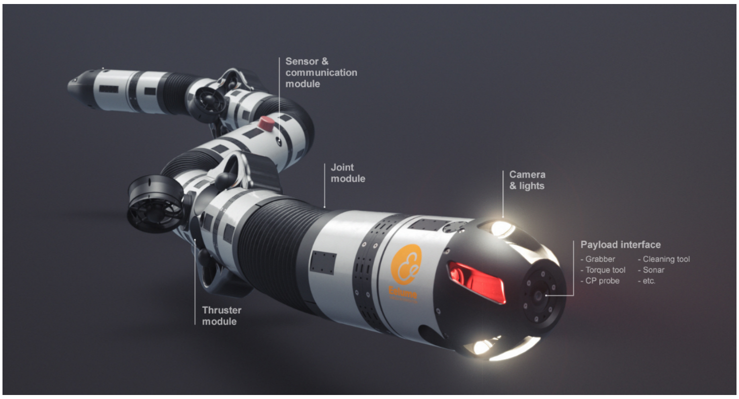
Kapittel 1

Introduction

The USM is an emerging technology that intends to reduce cost and increase versatility with respect to subsea inspection and intervention. Its slender and articulated body, see figure 1.1, allows access and maneuverability in confined and unstructured environments, which contrasts the more conventional remotely operated underwater vehicles (ROVs), see e.g. Wynn et al. (2014). Ideally, the USM is able to perform these tasks without any operator intervention. However, algorithms guaranteeing that the USM can navigate safely in a subsea environment and perform an arbitrary set of assignments are yet to be developed. A part of the solution is the design of a collision avoidance and control system that can interact with the navigation module and safely traverse the environment.

Both collision avoidance and control are studied, see e.g. Kohl et al. (2017), Kelasidi et al. (2014a), Borlaug et al. (2018) and Sverdrup-Thygeson et al. (2018), albeit the latter is more heavily emphasized in the context of USMs. Research on the former is motivated by the desire of safely avoiding collisions with newly discovered obstacles without violating the constraints imposed by the kinematic structure of the USM. Such a consideration is necessary for the control system to position the joints and actuate the thrusters to position the USM accordingly. Although modular control algorithms have been developed, see e.g. Sverdrup-Thygeson et al. (2018), Borlaug et al. (2018), Moe et al. (2016), with suitable properties, their composition into a complete system with an interface toward the collision avoidance module is non-trivial. When navigating the USM autonomously, the sensors, located strategically on the USM, require smooth motion and a position that maximize the performance of the navigation module. Generally, this objective doesn't comply with collision avoiding maneuvers which typically characterize as fast, nonlinear motion. The imposed complexity on the control system is experimentally demonstrated in Moe et al. (2016) for robot manipulators.

This project propose a design on how to compose a control system including guidance, collision avoidance and kinematic control from a practical perspective. Firstly, mathematical pre-requisites and assumptions are presented in chapter 2. A higher level view on path planning, with emphasis on path constraints are discussed in chapter 3. This chapter also discuss various approaches to collision avoidance (reactive path planning) and how to interface it with the path planning and guidance modules. In chapter 4, different guidance and kinematic controllers are proposed as constituent parts of the final control system. A dynamic controller used for simulations is also presented for completeness. Chapter 5 describe an experimental setup to validate the performance of the control system along with simulation results. Finally, conclusion and further work are provided in chapter 6.



Figur 1.1: Picture of a USM with thrusters and a camera module, by virtue of Eelume.

Kapittel 2

Preliminaries

This chapter will introduce notation that will be used in this project. Because the notation in the control systems and path planning literature differ, a joint notation is used following the standards of the USM community at NTNU, see e.g. Sverdrup-Thygeson et al. (2018). An extensive review of rigid body kinematics, differential kinematics and differential geometry is not provided, and more rigorous sources are therefore provided in the subsequent sections. This chapter firstly present basic kinematic transformations and conventions that are used in section 2.1. Then differential kinematics are presented in section 2.2 followed by the singularity robust task priority framework in section 2.3 to provide the foundation for the kinematic controller. The dynamic model used for modelling and simulations are presented in section 2.4. Finally, some basics on differential geometry and 3D curves are presented in section 2.5.

2.1 Kinematics

To represent a rigid bodies in 3D, the framework presented in From (2014) will be deployed.

2.1.1 Homogeneous transformations

Definition 2.1.1. A homogeneous transformation from frame F_j to from F_i will be represented by

$$g_{ji} = \begin{bmatrix} R_{ji} & p_{ji} \\ 0_{1\times 3} & 1 \end{bmatrix}, \ g : R^3 \longrightarrow R^3 \tag{2.1}$$

with inverse

$$g_{ij} = (g_{ji})^{-1} = \begin{bmatrix} R_{ij} & -R_{ij}p_{ji} \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{2.2}$$

Remark 2.1.1. *For all practical purposes $g_{ji} \in SE(3)$, i.e. g is a member of the group of rigid body transformations on R^3.*

Remark 2.1.2. *The origin of F_j with respect to F_i is denoted p_{ji}*

A useful result is that the composition of multiple homogeneous transformations is

$$g_{ji} = g_{j(j+1)}g_{(j+1)(j+2)} \cdots g_{(i-1)i} \tag{2.3}$$

A particular transformation deserving a presentation is the home transformation

Definition 2.1.2. The home transformation is the transformation from F_j to the origin of F_i without the rotation of F_i.

$$g_{j\bar{i}} = \begin{bmatrix} R_{j(i-1)} & p_{ji} \\ 0^T & 1 \end{bmatrix} \tag{2.4}$$

The transformation (2.4) are used extensively in the development of the differential kinematics in section 2.2.

2.1.2 DH-convention

Following the framework defined in Liljebäck et al. (2014), the translations and rotations between the links of a underwater mulit-body robot can be described using the dh-convention.

Definition 2.1.3. Assuming only revolute joints the homogeneous transformation matrix becomes

$$g_{i(i+1)}(q_i) = \begin{bmatrix} \cos(q_i) & -\sin(q_i)\cos(\alpha_i) & \sin(q_i)\sin(\alpha_i) & a_i\cos(q_i) \\ \sin(q_i) & \cos(q_i)\cos(\alpha_i) & -\cos(q_i)\sin(\alpha_i) & a_i\sin(q_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

The formulation in definition 2.1.3) simplifies the implementation between links of the robot considerably, making it a suitable choice for this project.

2.1.3 Euler convention

To represent the orientation of the robot of the robot relative to the inertial frame, Euler angles are used. This is to simplify debugging purposes, as they are *practically* much easier to comprehend than i.e. quaternions. The ZYX convention will be used giving a representation of the form

$$R_{0b} = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi c\phi s\theta \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{2.6}$$

where $s(\cdot)$ means $\sin(\cdot)$.

Remark 2.1.3. $\begin{bmatrix} \phi & \theta & \phi \end{bmatrix}^T$ *represents roll, pitch and yaw angles respectively.*

Remark 2.1.4. F_0 *denotes the inertial frame.*

Remark 2.1.5. *F_b denotes the body frame which has a fixed relation to somewhere on the robot.*

It should be noted that the Euler angles introduce a singularity when developing the mapping between the body velocities of the rigid body of the robot to the time derivative of the position variables. From chapter 3.3 in From (2014) it is derived that

$$
\begin{aligned}
\dot{\eta} &= J_b(\eta_{0b,2})V_{0b}^B \\
&= \begin{bmatrix} R_{0b}(\eta_{0b,2}) & 0 \\ 0 & J_{0b,\mathrm{rot}}(\eta_{0b,2}) \end{bmatrix} \\
&= \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta & 0 & 0 & 0 \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi c\phi s\theta & 0 & 0 & 0 \\ -s\theta & c\theta s\phi & c\theta c\phi & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \frac{s\phi s\theta}{c\theta} & \frac{c\phi s\theta}{c\theta} \\ 0 & 0 & 0 & 0 & c\phi & s\phi \\ 0 & 0 & 0 & 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix}
\end{aligned} \tag{2.7}
$$

Remark 2.1.6. *$\eta_{0b} = \begin{bmatrix} \eta_{0b,1}^T & \eta_{0b,2}^T \end{bmatrix}^T = \begin{bmatrix} x_{0b} & y_{0b} & z_{0b} & \phi & \theta & \psi \end{bmatrix}^T$ is the location of base link in space.*

Note from (2.7) that $J_b(\cdot)$ is undefined for $\theta = \pm\frac{\pi}{2}$.

2.2 Differential kinematics

This section will briefly present the necessary tools too solve the inverse kinematics problem, inspired by From (2014). The goal is to find a relationship between joint velocities, and linear and angular velocities of a chosen frame. Before defining this relationship, the adjoint map will be defined;

Definition 2.2.1. The adjoint map is defined as

$$\mathrm{Ad}_{g_{b\bar{i}}} = \begin{bmatrix} R_{b\bar{i}} & \hat{p}_{b\bar{i}} R_{b\bar{i}} \\ 0 & R_{b\bar{i}} \end{bmatrix} \tag{2.8}$$

with inverse

$$\mathrm{Ad}^{-1}_{g_{b\bar{i}}} = \begin{bmatrix} R_{\bar{i}b} & -R_{\bar{i}b}\hat{p}_{b\bar{i}} \\ 0 & R_{\bar{i}b} \end{bmatrix} \tag{2.9}$$

Remark 2.2.1.

$$\hat{x} = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{2.10}$$

for $x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \in R^3$.

To avoid excessive notation and definitions from From (2014), theorem 5.6 from this book will be presented. This theorem provides a relation between the body twist coordinates, V^B_{0i}, of some frame and the body twist coordinates of the base V^B_{0b} and joint velocities. The body velocity twist, V^B_{0i}, express the velocity of F_i with respect to F_0 as seen from F_i. Furthermore,

Definition 2.2.2. The USM pose $\xi_{0b} = \begin{bmatrix} \eta^T_{0b} & q^T \end{bmatrix}^T$ consists of the inertial pose of the base η_{0b} and all joint angles q. The USM velocities are expressed by $\zeta_{0b} = \dot{\xi}_{0b} = \begin{bmatrix} (V^B_{0b})^T & \dot{q}^T \end{bmatrix}^T$.

The theorem can now be stated;

Theorem 2.2.1. *The body geometric Jacobian* $\bar{J}^B_{gi}$ *that gives the relation* $\bar{V}^B_{o\bar{i}} = \bar{J}^B_{gi}(q)\zeta$ *is given by*

$$\bar{J}^B_{gi}(q) = \begin{bmatrix} Ad^{-1}_{g_{g\bar{i}}} & Ad^{-1}_{g_{g\bar{i}}} J_i(q) \end{bmatrix} \in R^{6\times(6+n)} \tag{2.11}$$

such that

$$V^B_{0\bar{i}} = \bar{J}^B_{gi}\zeta_{0b} \tag{2.12}$$

Remark 2.2.2. *The manipulator Jacobian $J_i(q)$ in (2.11) is a Jacobian without the influence of the last joints*

$$J_i(q) = \begin{bmatrix} Ad_{g_{b\bar{1}}} X_1^1 & Ad_{g_{b\bar{2}}} X_2^2 & \cdots & Ad_{g_{b\bar{i}}} X_i^i & 0_{(n-i)x6} \end{bmatrix} \tag{2.13}$$

where the body joint twists, or the direction of allowed motion as seen from the joint, are expressed as

$$X_{i,\,revolute}^i = \begin{bmatrix} v_i^i \\ \omega_i^i \end{bmatrix} = \begin{bmatrix} 0 \\ p_i^i \end{bmatrix} \tag{2.14}$$

under the assumption of revolute joints.

In particular, the mapping from base velocities to end-effector velocities is important for the USM to be controlled with respect to the leading end-effector. Given the transformation from the base link to the leading end-effector is given by g_{be}, the velocity of the end-effector is

$$V_{0\bar{e}}^B = \begin{bmatrix} Ad_{g_{g\bar{i}}}^{-1} & Ad_{g_{g\bar{i}}}^{-1} J_i(q) \end{bmatrix} \begin{bmatrix} V_{0b}^B \\ \dot{q} \end{bmatrix} \tag{2.15}$$

2.3 SRMTP

Chiaverini (1997) presents a framework for prioritizing multiple kinematic tasks with a redundant manipulator arm. This framework can be extended to USM as demonstrated in Sverdrup-Thygeson et al. (2017). This section will describe this framework.

2.3.1 Single task

Tasks describe variables or some mathematical quantity (consisting of the same variables) of the mechanical system that the user want to control, i.e. the position of the end-effector. In general, such a task is given by

$$\sigma = f(\eta_{0i}, q) \tag{2.16}$$

Example 2.3.1. *The end-effector task is*

$$\sigma_{pos,i} = \begin{bmatrix} x_{0e} & y_{0e} & z_{0e} \end{bmatrix}^T \tag{2.17}$$

Remark 2.3.1. *By example 2.3.1 it can be seen that* σ *can be a vector function.*

Remark 2.3.2. *Tasks will inherit the subscript used in example 2.3.1, with pos describing the type of task and i denoting which coordinate frame the task is related to.*

Differentiating equation (2.16) with respect to the configuration variables yields

$$\dot{\sigma} = J_i(\eta, q)\zeta \tag{2.18}$$

where J_i represents the task Jacobian.

Remark 2.3.3. *Considering example 2.3.1, the task Jacobian can be expressed as*

$$J_i = J_e = \bar{J}^B_{ge}(q) \tag{2.19}$$

That is, the body geometric Jacobian from equation (2.11).

Since the overall goal of the method is to compute reference velocities ζ_{0i} for some coordinate frame. Computing the inverse relation of equation (2.18 requires the introduction of the pseudo-inverse,

$$J_i^\dagger = J_i^T (J_i J_i^T)^{-1} \tag{2.20}$$

This can also be computed in a variety of different formats, detailed in Moe et al. (2016), e.g. using a weighted pseudo-inverse, equation (2.21)

$$J_i^\dagger = W^{-1} J_i^T (J_i W^{-1} J_i^T) \tag{2.21}$$

The following expression can be developed according to the desire of reducing the error between the velocity of the base and the joint angle and their respective references to zero

$$\zeta_{\text{ref}} = J_i^{\dagger}(\dot{\sigma}_d + \Lambda\tilde{\sigma}) \tag{2.22}$$

where $\tilde{\sigma} = \sigma_d - \sigma$ and Λ_i is a positive definite matrix of gains, Moe et al. (2016).

Remark 2.3.4. *Note that the reference ζ_{ref} specifies the desired base velocities in the inertial frame and all joint angles relative to the home transformations used in the calculation of the geometric Jacobian (2.11).*

2.3.2 Multiple tasks

The extension of equation (2.22) to the multiple task case requires exploiting that the null-space of the task Jacobian is nonempty. That is, the projection matrix N_i onto the null space of J_i has non-zero rank

$$N_i = (I - J_i^T J_i) \tag{2.23}$$

This is explained in Siciliano et al. (2009) as attempting to satisfy additional constraints without violating the constraint (2.22). Generally, there is a plethora of tasks that can be added, expressed according to Moe et al. (2016)

$$\zeta_{\text{ref}} = J_1^{\dagger}(\dot{\sigma}_{1,d} + \Lambda_1\tilde{\sigma}_1) + N_1 J_2^{\dagger}(\dot{\sigma}_{2,d} + \Lambda_2\tilde{\sigma}_2) + \ldots + N_{12\ldots(k-1)} J_k^{\dagger}(\dot{\sigma}_{k,d} + \Lambda_k\tilde{\sigma}_k) \tag{2.24}$$

where Λ_i is gains similar to in equation (2.22) and $N_{1,\ldots j} := (I - J_{12..j}^T J_{12..j})$ is the common null space for the tasks 1 to j. $J_{12..j}$ then represents a stacked Jacobian following the definition

$$J_{12..j} = \begin{bmatrix} J_1 \\ J_2 \\ . \\ . \\ J_j \end{bmatrix} \tag{2.25}$$

2.4 System model

The dynamic model can be expressed as Borlaug et al. (2018), inspired by From (2014)

$$M(q)\dot{\zeta}_{0b} + C(q, \zeta_{0b})\zeta_{0b} + D(q, \zeta_{0b})\zeta_{0b} + G(\xi_{0b}) = \tau(q) \tag{2.26}$$

where $\zeta = \begin{bmatrix} \dot{\eta}_{0b}^T & \dot{q}^T \end{bmatrix}^T$ is the base and joint velocities, $M(q)$ is the inertia matrix including added mass terms, $C(q, \zeta_{0b})$ is the Coriolis-centripetal matrix, $D(q, \zeta_{0b})$ is a damping matrix, and $G(\xi_{0b})$ is the matrix of gravitational and buoyancy forces. In addition, a kinematic model is required to express the model with respect to an inertial frame

$$\dot{\xi}_{0b} = J(q)\zeta_{0b} \begin{bmatrix} R_{0b} & 0_{3\times 3} & 0_{3\times(n-1)} \\ 0_{3\times 3} & J_{0b,\mathrm{rot}}^{-1} & 0_{3\times(n-1)} \\ 0_{(n-1)\times 3} & 0_{(n-1)\times 3} & I_{(n-1)\times(n-1)} \end{bmatrix} \tag{2.27}$$

where g_{0b} is the inertial orientation of the base link expressed with the xyz-euler convention from section 2.1.3 and $J_{0b,\mathrm{rot}}$ is the inertial Jacobian defined in (2.7).

Remark 2.4.1. *n in equation (2.27) denotes the number of links on the USM implying that $n - 1$ is the number of joints. This notation will be used henceforth.*

2.5 Differential geometry

Spatial curves can be generated in a number of ways, Bezièr curves, B-splines, natural splines, Hermite splines, etc. Results from Tapp (2016) on 3D curves used in this project are presented in this section without proofs. Firstly, natural splines are used in this project due to their simplicity and because they interpolate all waypoints exactly. This choice does not reflect any practical experiment and does not devalue the practical usefulness of other representations.

Definition 2.5.1. A parametrized curve in $\mathbb{R}^n$ is a smooth function $p : T \longrightarrow \mathbb{R}^n$, where $T \subset \mathbb{R}$ is an interval.

Remark 2.5.1. *In 3D definition 2.5.1 implies, provided sufficient smoothness, that the position, velocity and acceleration vectors of the curve can be expressed by*

$$\begin{aligned} p(t) &= (x(t), y(t), z(t)) \\ v(t) &= (x'(t), y'(t), z'(t)) \\ a(t) &= (x''(t), y''(t), z''(t)) \end{aligned} \tag{2.28}$$

For convenience, it is desirable to work with an orthonormal collection of vectors in 3D. This collection is composed by using that the acceleration can be decomposed into a part parallel $a_{||}$ and orthogonal $a_{\perp}$ to the velocity vector;

$$a(t) = a_{||}(t) + a_{\perp}(t) \tag{2.29}$$

The rate of change of the velocity can therefore be found computed as

$$\frac{d}{dt}v(t) = \frac{\langle a(t), v(t) \rangle}{v(t)} \tag{2.30}$$

Based on equations (2.29) and (2.30), the following definition will be used to define an orthononormal basis

Definition 2.5.2. Let $p(t) : I \longrightarrow,\ t \in I$ be a regular curve. Define the **unit tangent** and the **unit normal** vectors at t as

$$\begin{aligned} t(t) &= \frac{v(t)}{|v(t)|} \\ n(t) &= \frac{a_{\perp}(t)}{|a_{\perp}(t)|} \end{aligned} \tag{2.31}$$

In addition, the following proposition is useful for computing the orthogonal part of the acceleration vector in a practical way

Proposition 2.5.1. *Consider the derivative of the unit tangent vector at* t*,* $t'(t)$*. The orthogonal part of the acceleration vector can be defined as*

$$a_{\perp}(t) = |v(t|t'(t) \tag{2.32}$$

which implies that the unit normal vector can be defined as

$$n = \frac{a_{\perp}(t)}{|a_{\perp}(t)|} = \frac{t'(t)}{|t'(t)|} \tag{2.33}$$

To establish further necessary properties of the path, the curvature and the torsion can be defined in terms of the acceleration and acceleration vectors.

Definition 2.5.3. In 3D, the **curvature** of a curve is defined as

$$\kappa = \frac{||v(t) \times a(t)||}{||v(t)||^3} \tag{2.34}$$

and the **torsion** as

$$\tau = \frac{\det(p(t), v(t), a(t))}{||v(t) \times a(t)||^2} \tag{2.35}$$

Remark 2.5.2. *Both curvature and torsion require constant time to evaluate for a polynomial spline.*

Definition 2.5.4. Let $p(t) : I \longrightarrow, \ t \in I$ be a regular curve. Further assume that $\kappa(t) \neq 0$. The **Frenet frame** at t is the orthonormal basis $t(t), n(t), b(t)$ of $\mathbb{R}^3$ defined as

$$
\begin{aligned}
t(t) &= \frac{v(t)}{|v(t)|} \\
n(t) &= \frac{t'(t)}{|t'(t)|} \\
b(t) &= t(t) \times n(t)
\end{aligned}
\tag{2.36}
$$

Individually they are called the **unit tangent**, **unit normal** and **unit binormal** vectors at t.

The torsion can now be re-defined

Definition 2.5.5. Let $p(t) : I \longrightarrow,\ t \in I$ be a regular curve. Further assume that $\kappa(t) \neq 0$. The **torsion** of $p(\cdot)$ at t is

$$
\tau(t) = \frac{-\langle b'(t), n(t) \rangle}{|v(t)|} \tag{2.37}
$$

For the purpose of guidance and kinematic control the closest point between the curve $p(t)$ and the position of the USM, $\eta_{0b,2}$, is required. In the general case, this is equivalent to solving the problem

$$
t^* = \text{argmin}_t \ (\eta_{0b,2} - p(t))^2 \tag{2.38}
$$

which again is equivalent to finding the orthogonal projection of the USM position onto the curve, meaning $t^* = t_\perp$. Solving this problem can be done efficiently using Newton iterations from a start point sufficiently close to the origin. The details behind the implementation are not of importance to the results in this project and are therefore left out. It should be noted, however, that the problem (2.38) is non-convex which may introduce strange results if not treated carefully.

Kapittel 3

Planning

Path planning is the problem finding a feasible path between an initial state η_0 to a goal state η_d in finite time. Furthermore, the work space is $\mathbb{R}^3$ making the initial problem more difficult than in $\mathbb{R}^2$ LaValle (2006), Siciliano et al. (2009). Firstly, a definition of the configuration space in $\mathbb{R}^3$ is necessary to describe the problem

Definition 3.0.1. The configuration space C in $\mathbb{R}^3$ is $\eta = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T \in \mathbb{R}^6$.

Definition 3.0.1 implies that the general problem has to be solved for 6 degrees of freedom. Additionally, the objective can be made more difficult by further considering the globally shortest path. This project will not consider such an objective, but rather local planning to determine a feasible path that satisfy certain criteria. However, the overall planning system will be presented for completeness and to introduce the required further work towards a first working prototype of a deliberative architecture for USM control. The full architecture draws inspiration from the architecture presented in McGann et al. (2008) and the goal-oriented T-REX system Rajan et al. (2000). Whence, considerations on timing issues related to re-planning and reactive collision avoidance will be considered, but a complete elaboration on the architecture is out of the scope of this project.

The reason for firstly prioritizing the collision avoidance system is from a safety perspective and to motivate the definition of feasibility criteria necessary to control the USM safely.

Remark 3.0.1. *The term reactive path planning is adopted from this point on.*

This chapter will firstly review all definitions and assumptions required to ensure a proper definition of feasibility in the context of USM path following. Then a literary survey of collision avoidance approaches applied to USMs and other 3D methods of interest, not yet applied, will be presented. Finally a complete architecture for the design of the interface between deliberative higher level path planners and the collision avoidance system will be discussed.

3.1 Path criteria

Reactive path planning intend to construct a local collision free path that is feasible for path following. Firstly the problem of feasibility will be treated. That is, the path has to obey certain kinematic restrictions on curvature and torsion imposed by the configuration of the USM while avoiding collisions.

Definition 3.1.1. If the constraints $\kappa_p \leq k_\kappa$ and $\tau_p \leq k_\tau$ are satisfied, where $\kappa_p, \tau_p \in \mathbb{R}$ is the curvature and torsion of the planned path and $k_\kappa, k_\tau \in \mathbb{R}$ are their limits imposed by the USM configuration, then the path is considered kinematically feasible.

In 2D, a curvature criteria can be developed by wrapping the USM around a circle, see figure 3.1. Consider a single link with length l_i, stretched as a segment across a circle with the joints at each end of the link touching the circle. Denote the radius of the circle by R and denote the angle θ_i as the intermediate angle between the two legs of the equilateral triangle kissing the link l_i at both ends. By figure 3.1

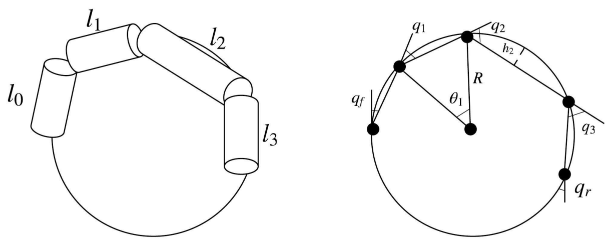
$$\frac{l_i}{\sin(\frac{\theta_i}{2})} = \frac{2R}{\sin(90)} \tag{3.1}$$

Whence,

$$\kappa_i = \frac{2\sin(q_i)}{l_i} \tag{3.2}$$

since $\phi = \pi - q_i \equiv \frac{\theta_i}{2}$.

To develop a hard criteria, however, the offset from the previous link has to be taken into account. From figure 3.1 it can be seen that the offset is equivalent to the joint angle of the previous link q_i. It is therefore proposed that the USM is limited to follow a circle, *perfectly*, when one of the joints deviate physically from the circle.



Figur 3.1: To the left, a image of the links of the USM wrapped around a circle. To the right, the joint angles q_i (with front and rear end-effector q_f, q_r) inscribed to visually demonstrate the propagated angular offset from the previous link.

Proposition 3.1.1. *A USM with links of lengths l_i and joint angles q_i, with maximum joint angle $q_{i,max}$. The path curvature κ has to satisfy the following set of constraints*

$$
\begin{aligned}
q_1 &= \arcsin\left(\frac{l_1}{2\kappa}\right) + q_f \;,\; \frac{\theta_1}{2} \leq q_{1,\,max} - q_f \\
q_2 &= \arcsin\left(\frac{l_2}{2\kappa}\right) + \frac{\theta_1}{2} \;,\; \frac{\theta_2}{2} \leq q_{2,\,max} - \frac{\theta_1}{2} \\
&. \\
&. \\
q_n &= \arcsin\left(\frac{l_n}{2\kappa}\right) + \frac{\theta_{n-1}}{2} \;,\; \frac{\theta_n}{2} \leq q_{n,\,max} - \frac{\theta_{n-1}}{2}
\end{aligned}
\tag{3.3}
$$

When the curvature exceed one of these limits, it is kinematically impossible for the USM to follow the circular path.

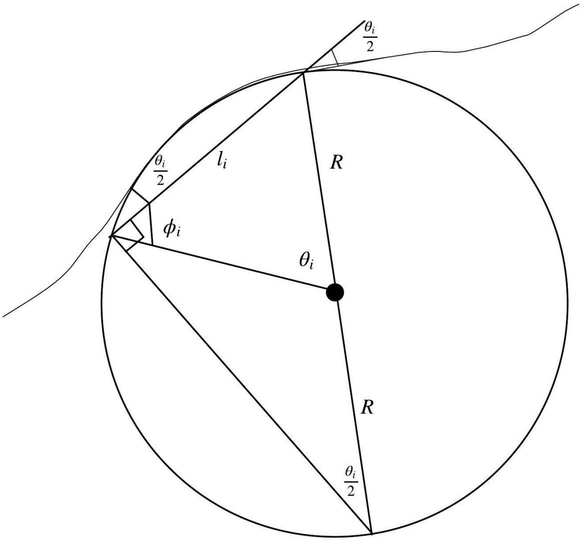
Bevis. Start with expression (3.2) for joint q_0 and figures 3.1 and 3.2 describing the propagating angular offsets. Thus $\frac{\theta_1}{2} \leq q_{1,\mathrm{max}} - q_f$ has to be satisfied for the first joint to cut with the circle segment. Whence, that $q_f + \frac{\theta_1}{2} = q_1$. From figure 3.2 $\frac{\theta_1}{2}$ will propagate implying that the next joint angle have to satisfy $\frac{\theta_2}{2} \leq q_{2,\mathrm{max}} - \frac{\theta_1}{2}$. Repeating this argument until the last joint completes the proof. □

Remark 3.1.1. *The argument extends to non-circular paths, assuming availability of the maximum curvature of the path enclosed by the link, see figure 3.2. The argument in proposition 3.1.1 can then be approximated with $\kappa = \kappa_{i,\,max}$ determined by an over-estimate of the joint angle.*

Having established the joint limits relative to the maximum curvature of a path, it is interesting to analyze the path deviation. There are two types of deviation; path deviation due to cutting the curve and path deviation due to exceeding joint limits.

Corollary 3.1.1. *Assuming that the joint limits (3.3) is satisfied, the maximum path deviation is, see figure 3.1,*

$$h_i = R - \sqrt{R^2 - \frac{l_i^2}{4}} \tag{3.4}$$



Figur 3.2: The curved path with a circle drawn at the point of maximum curvature. The angles $\frac{\theta_i}{2}$ shows that the link cutting the circle over-estimates the actual joint angle necessary. The extended line across the circle additionally provide the geometric relations necessary to show how the angles propagate.

Remark 3.1.2. *When* $R >> l_i$ *the path deviation goes to zero.*

Remark 3.1.3. *The above argument is not directly applicable to 3D, because of the interaction between joint rotations in yaw and pitch around a sphere. It can however provide an approximate limit for practical purposes.*

Considering related work, Liljebäck et al. (2014) discuss how a dynamic motion pattern can be generated based on a continuous curve. It does not however discuss the explicit constraints on the path. Kelasidi et al. (2014a) and Kohl et al. (2017) propose methods for 2D collision avoidance for USMs moving in a undulating pattern. They propose an additional region around the snake robot and the obstacles to avoid collisions due to the oscillating motion of the USM. This additional safe distance can be reduced, however, the path deviation (3.4) have to be taken into account.

The problem of constructing a collision free path in an a priori unknown and unstructured environment is presented following the notation in LaValle (2006).

Definition 3.1.2. $C_{\text{obstacles}} \cap C_{p,i} = \emptyset$ for all $\bar{C}_{p,i}$ queried from $p(\cdot)$ where i denote a particular waypoint along the path $p(\cdot)$, $C_{\text{obstacles}} \in \mathbb{R}^3$ the spatial extent of all obstacles in the work space of interest and $C_{p,i} \in \mathbb{R}^3$ the spatial position of a particular waypoint.

The relation between curvature, link length and path deviation in (3.4), definition 3.1.2 has to be augmented.

Addendum 3.1.2.1. Define $h_{\max}$ as the maximum deviation $h_{i,\max}$ from the path when wrapping the USM around a circle of maximum possible curvature. A set of circular cross-sections of radius $h_{\max}$ centered at each waypoint $C_{p,i}$ transverse to the direction of the path can now be defined, say $\bar{C}_{p,i}$, such that $C_{\text{obstacles}} \cap \bar{C}_{p,i} = \emptyset$ for all $\bar{C}_{p,i}$ queried from $p(\cdot)$.

Remark 3.1.4. *Note that $p(\cdot)$ is a polynomial path interpolated between a set of waypoints in $\mathbb{R}^3$, implicitly representing the desired pose of the USM.*

A collision avoiding path can thereby be described as a set of cross-sections, located sufficiently close, such that none of them intersects any obstacles. They have to be located sufficiently close for two reasons; no obstacles can fit between any two waypoints and the curve resulting from the polynomial interpolation must not deviate to much from the straight line between any two waypoints. Bèzier curves or Fermat spirals are therefore considered unfit for polynomial interpolation for this application (they will at a minimum require more work to guarantee collision avoidance). More obvious choices are natural cubic splines, cubic Hermitè splines, or similar approaches. In this work, natural cubic splines will be used for their simplicity as mentioned in section 2.5.

When working with USMs, it is important to remember that they are kinematically configured with long, slender and articulated bodies. It is therefore not trivial to define what part (or multiple parts) of the body the path represents. The most intriguing choice is to have the path represent the head of the USM and ensure collision avoidance by maintaining the rest of the body as close to track as possible. However, it may not be desirable in some cases as the USM may want to perform inspection tasks, or use the sensors placed on the head for some other task. In such cases, it may be desirable to have the path represent an internal link on the USM and have the head free to perform the necessary task. Further, some USMs can be configured to have a specific link that is longer than the others and thus more suitable as a base link. However, these questions are more related to control than planning and will therefore be discussed thoroughly in chapter 4. For now it is assumed that the path represents some part of the USM and that the rest of the body is maintained as close to the track as possible.

All the above considerations can be summarized in the following list of planning criteria

- The path is planned for some link on the USM under the assumption that the body of the USM follows, see e.g. Liljebäck et al. (2014).
- The path must be feasible considering the kinematics of the USM according to definition 3.1.1.
- The path must ensure collision avoidance for all links according to definition 3.1.2.

A final measure that will we considered is the continuity of the resulting path, in particular continuity in curvature and torsion as it un-desirable to control the USM in a jerky motion pattern.

Remark 3.1.5. *A curve* $p(\cdot) \in C^2$ *has continuous curvature.*

Remark 3.1.6. *A curve* $p(\cdot) \in C^3$ *has continuous torsion.*

By remarks 3.1.5 and 3.1.6, cubic polynomials are desirable as a minimum smoothness criteria.

Further considerations to take into account is the density of the map received from the mapping algorithm. This map may contain anything from the essential features defining a obstacle, like their center and a circular bounding radius, to dense grid maps portraying even their rugged surfaces. At this point, no assumptions are made regarding the input to the reactive path planning algorithm. The algorithms will be discussed on a general basis and their inherent assumptions about the environment will be clearly stated.

3.2 Reactive path planning

Reactive path planning is constrained by the geometric criteria in section 3.1 to hold in addition to hard run-time constraints. It is a low level path planner which is a part of the AUV executive layer, see figure 3.3, and is therefore required to have good real-time performance in addition to strict feasibility and collision avoidance criteria. In practice, it is undesirable to constantly stop as an effect of slow planning because of efficiency and safety. Efficiency is a factor because the USM may have to counteract motion currents which require an increased amount of electrical power to maintain its systems running. Safety is a factor due to the possibility of collisions. If the system reacts to slow, collisions may be unavoidable.

This section will first present a survey of reactive methods deployed in the USM community emphasizing approaches already deployed on USMs, see Kohl et al. (2017), Kelasidi et al. (2014a) and Hoffmann (2018), and promising approaches successfully applied in related 3D scenarios. Examples come from aerial unmanned vehicles Belkhouche and Bendjilali (2012), Schøler (2012) and underwater autonomous vehicles Wiig et al. (2018) to mention a few.

3.2.1 Artificial potential fields

The algorithm proposed in Kelasidi et al. (2014a) is based on artificial potential fields (APFs) as a first step in a motion planner. The algorithm is inspired by Khatib (1986) which proposed the APF approach as an idea for making obstacles repel the robot and making the end goal attract it. To account for the volume of the robot and obstacles a potential having a maximum at the obstacle boundaries and a minimum at a boundary around the robot is defined. Note that the boundary around the robot has direct influence on the steepness of the curve and is therefore used to account for the joint constraints of the robot. The feasibility criteria defined in section 3.1 are therefore implicitly encoded in the boundary potentials. Since interpolating waypoints are computed along the derived path from the APF planner, a interpolating spline can quickly be computed according to the desired criteria. In Kelasidi et al. (2014a) it is further proposed that a harmonic potential field can be chosen to avoid local minima.

In chapter 8.4 in LaValle (2006) harmonic potential functions are described as satisfying the differential equation

$$\Delta^2 U = \sum_{i=1}^{\dim(WS)} \frac{\partial^2 U}{\partial x_i^2} = 0 \tag{3.5}$$

It is formally defined with some choice of boundary conditions on the state space which constrain the boundary to maintain a constant value (Dirichlet boundary conditions). The downside with this approach is that the obstacle boundaries are not constructed explicitly, in general, (this is an artifact of collision checking, see e.g. chapter 5.3 in LaValle (2006)) and that it is hard to compute any numerical solution. However, value iteration can be applied to obtain an approximate solution, but this too implies a heavy computational burden which is not desirable in the design of a reactive planning algorithm. The APF algorithm also suffer from oscillating behaviour which is not desirable from a control perspective.

3.2.2 Task based avoidance

A task based approach applied to collision avoidance was introduced by Moe and Pettersen (2016) with surface vessels. In Kohl et al. (2017) a 2D collision avoidance algorithm for USMs exploiting the inherent properties of a task based framework was presented. The algorithm is based on defining two modes of operation; a path following mode and a obstacle avoidance mode. The path following mode constitutes any choice of guidance algorithm (see section 4.2 for an example), whereas the obstacle avoidance mode relies on the definition of a set-based task measuring the distance between a desired set of points on the USM and an obstacle.

Definition 3.2.1. The set-based obstacle avoidance tasks can be defined as

$$\sigma_{\mathrm{CA},\,i} = \sqrt{(x_{0i} - x_{0,\mathrm{Obs}})^2 + (y_{0i} - y_{0,\mathrm{Obs}})^2 + (z_{0i} - z_{0,\mathrm{Obs}})^2)} \tag{3.6}$$

with its derivative

$$\dot{\sigma}_{\mathrm{CA},\,i} = \frac{(x_{0i} - x_{0,\mathrm{Obs}})(\dot{x}_{0i} - \dot{x}_{0,\mathrm{Obs}}) + (y_{0i} - y_{0,\mathrm{Obs}})(\dot{y}_{0i} - \dot{y}_{0,\mathrm{Obs}}) + (z_{0i} - z_{0,\mathrm{Obs}})(\dot{z}_{0i} - \dot{z}_{0,\mathrm{Obs}})}{\sqrt{(x_{0i} - x_{0,\mathrm{Obs}})^2 + (y_{0i} - y_{0,\mathrm{Obs}})^2 + (z_{0i} - z_{0,\mathrm{Obs}})^2)}} \tag{3.7}$$

The framework proposed by Moe and Pettersen (2016) relies on turning the task into an equality based task if it is about to leave its valid set

$$\begin{aligned} D &:= [\sigma_{\min}, \sigma_{\max}] \\ &:= [\min(R_m, \max(\sigma_{\mathrm{Ca},\,i}, R_0)), \infty] \end{aligned} \tag{3.8}$$

where R_0 is the sphere radius around the obstacle and R_m is a safe radius composed onto R_0. This condition can be defined by the tangent cone, see section 2.3, the space of *desirable* directions of motion.

The upsides of this approach is its simplicity and computational efficiency. It is much quicker to compute compared to the APF method from section 3.2.1 and avoids the oscillation problems due to the defined safe radius R_m. The downside of this algorithm is due to the spherical representation of the obstacles. When approaching a sphere straight ahead, the USM will move directly towards it for so to start turning to either side when approaching the boundary radius. Since the USM is nonholonomic this won't be any problem to achieve in practice, but it may induce an increased burden on the thrusters along with rapid motions for the guiding link. Since cameras are placed on the guiding link, it is desirable for the purpose of navigation and mapping that its motion is smooth and predictable. This problem will partly be accounted for in the methods described in the subsequent sections.

The task based avoidance methodology satisfy the criteria from section 3.1 in a similar way as the APF approach in section 3.2.1. The feasibility of curving around an obstacle is implicitly defined by the bounding curve around the obstacle. Collision avoidance and safe passage can by this assumption be proven, see Kohl et al. (2017) and Moe and Pettersen (2016). However, it doesn't take into account the possibility of closely spaced obstacles defining an un-feasible passage as discussed in Hoffmann (2018). This is because of the inherent local scope of the algorithm.

3.2.3 Spherical avoidance

The algorithm proposed in Wiig et al. (2018) introduce an extended 3D vision cone which defines the set of safe heading and pitch angles (a similar approach was used in Belkhouche and Bendjilali (2012)). It also defines a criterion for transitioning between guidance mode and collision avoidance mode. Due to the properties of 3D space, a continuum of possible safe heading and pitch angles exist, and thus, an optimal criteria can be defined to choose an optimal heading pitch pair. In particular, the algorithm constructs a spherical boundary around a nearby object and finds the apex angle $2\gamma_a$. Additionally, the apex angle is augmented with a safety boundary $2\alpha_0$ to construct an augmented apex $2\gamma_e$ around the sphere to ensure safe passage

$$\begin{aligned} \gamma_e &= \gamma_a + \alpha_0 \\ &= \arcsin\left(\frac{R_0}{R_0 + d_0}\right) + \alpha_0 \end{aligned} \tag{3.9}$$

where d_0 is the distance from the robot to the sphere. Having defined the apex angle, a cost function C can be constructed to define the optimization problem to find the apex that introduce the least possible change of orientation for the robot

$$\phi^* = \arg\min_{\phi} C(\phi) \tag{3.10}$$

where $\phi \in \left[0, 2\pi\right]$ is the angular increment around the horizon of the sphere.

Collision avoidance and safe passage is proven in Wiig et al. (2018). The bounding sphere is her have to satisfy the criteria defined in 3.1. This is because the USM follow the the boundary during collision avoidance mode between the entering and exiting apexes. Finally, note that the case of multiple obstacle collision avoidance is not accounted for with this approach.

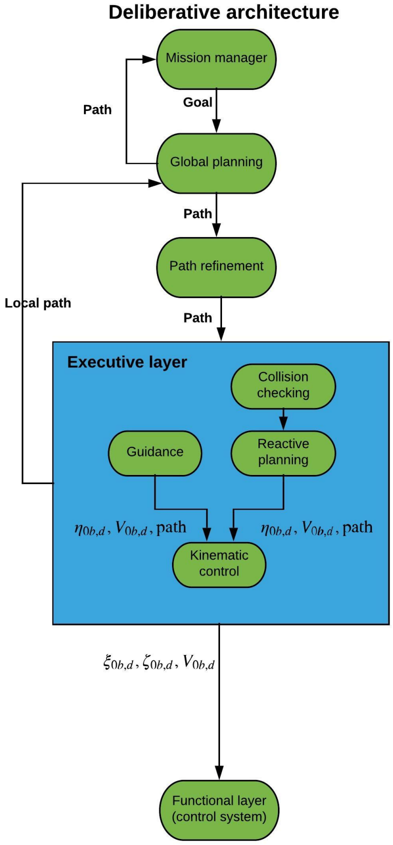
3.2.4 Geodesic planning

Schøler (2012) presents an approach for computing the shortest path between two points, possibly obstructed by some non-convex obstacle, in a 3D work space. Conceptually, the approach patch the obstructing obstacles with spheres, cylinders and planes, producing a convex boundary around it. A path composed of a straight line from the start point to the horizon of the obstacle, traveling the shortest way between the initial and target horizon and a straight line between the target horizon and the goal point is computed. This path is computed by solving a convex optimization problem, which was shown in Schøler (2012) to produce a collision avoiding path in reasonable time. In addition, feasibility can be assured by scaling the sphere and cylinder patches encapsulating the obstacle according to the USM configuration.

The upside of geodesic path planning over the approach presented in section 3.2.3 is that a more general set of obstacles can be traversed. The downside is that the approach doesn't scale well when obstacles are partially observed. Partially gained new information implies that a new convex boundary must be computed which in turn requires the optimization problem to be solved over again. The method may therefore introduce to much computational strain as a reactive path planner when approaching a previously unexplored obstacle.

3.3 Deliberative architecture

A survey of methods for more global path planning will not be discussed in this chapter and is a topic for further work. Nevertheless, the full planning interface will be presented because of its relevance towards implementing a deliberative system able to handle the tasks presented in chapter 1. In relation to McGann et al. (2008), the following layers are present in the USM agent, see figure 3.3; mission manager, global planning and path refinement and the executive.


Deliberative architecture
Mission manager
Goal
Path
Global planning
Path
Path refinement
Local path
Path
Executive layer
Collision checking
Guidance
Reactive planning
$\eta_{0b,d}, V_{0b,d}$, path
$\eta_{0b,d}, V_{0b,d}$, path
Kinematic control
$\xi_{0b,d}, \zeta_{0b,d}, V_{0b,d}$
Functional layer (control system)

Figur 3.3: Overview of the deliberative architecture with an interface between the mission manager and the path planners.

These layers have different scopes in terms of internal data, external interfaces and temporal requirements, and they require synchronization when sharing this data. In the T-REX framework these differences are classified with functional and temporal scopes on the various layers along with timing requirements;

- Functional scope: State variables of concern for deliberation and action.
- Temporal scope: Look-ahead window over which to deliberate.
- Timing requirements: Latency for a layer to deliberate on goals in its planning horizon.

This project will provide a definition of the various layers and more concretely define the requirements for the executive layer. The final prototype will define all the parameters for each of the layers in figure 3.3 and their interfaces with the computer vision and navigation modules.

3.3.1 Mission manager

The system is designed to complete missions including;

- Following straight lined and curved pipes.
- Waypoint guidance with user specified waypoints.
- Classifying a docking station and navigating towards it.

The missions will in the context of an initial working prototype be specified by the user and the mission manager is responsible for keeping track of the current mission status. Feedback is then provided from the global path planner as illustrated in figure 3.3, and the mission manager can define whether we deviate from the defined mission. Extensions such as sparse or dense mapping of the environment or an inspection mission can easily be added. These extensions include interfacing with the navigation and mapping modules in addition to the planning layers. They are therefore considered out of scope for this project.

3.3.2 Global planning and path refinement

The global path planning layer has the responsibility of maintaining the overall desired path of the USM. It has to compute a 3D path connecting an initial state with a desired goal state. This path will then be sent back to the mission manager to evaluate the mission status and to the path refinement module for checking compliance with the criteria in section 3.1. Whether this should be done hierarchically or in a single layer has not been evaluated and is subject to further work. Furthermore, it is assumed that these layers obtain feedback from the executive layers in addition to input from the navigation and mapping modules. Hence, it has sufficient information to perform extensive planning in a time window that spans seconds to minutes depending on the immediate urgency. The implementation and evaluation of the exact details in these layers is also subject to further work.

3.3.3 Executive layer

The executive layer obtains a global path and has the goal of computing a desired pose $\eta_{0b,d}$ and velocity $V_{0b,d}$ for the base link of the USM. The layer will also communicate directly with the mapping module such that an updated map is available at all times. The executive layer works in two separate threads defining whether the system is in path following mode or collision avoidance mode. The main thread, the guidance block in figure 3.3, is the guidance mode (various guidance algorithms are described in section 4.2) which compute references for the USM to steer it toward and along the provided path. The second thread, the collision checking and reactive planning blocks in figure 3.3, includes *collision checking* and reactive planning. Collision checking includes verifying whether the provided path collides with any obstacles within some distance around the USM. If a collision is detected, which may occur due to the assumed latency in the global planning layer, the system will enter collision avoidance mode and the reactive planner will guide the USM safely around the obstacles. This architecture is similar to the ones presented in Kohl et al. (2017) and Moe and Pettersen (2016) with regards to the division of path following and collision avoidance into disjoint modes.

Proposition 3.3.1. *Collision checking has run-time complexity O(NM), in the number of waypoints N and surrounding obstacles M, provided closed form expressions of the obstacles.*

Bevis. Consider a 3D path densely represented by a spline. A dense set of waypoints can be queried from the path in constant time $O(1)$, per waypoint, implying $O(N)$ run-time to query the full set. Under the assumption of a local area of interest around the USM filled with M obstacles represented by closed form expressions, collision checking for each waypoint can be performed in $O(M)$ time. Whence, the collision checking of all waypoints against all obstacles takes $O(NM)$ time. □

Remark 3.3.1. *Further assuming that the local area of interest is small and that obstacles are sparsely distributed in this area implies that* $M << N$, *hence the run-time complexity is reduced to approximate linear time O(N) in the number of waypoints.*

By proposition 3.3.1, it is desirable to represent obstacles with closed form expressions to minimize the time spent in collision checking (for the more elaborate general case without a closed-form representation see chapter 5 in LaValle (2006)). This will reduce the computational burden on the executive layer and reduce total run-time toward linear time deliberation and constant- or linear-time execution dependent on the reactive planner.

With regards to the choice of reactive planner, the properties of the spherical avoidance approach described in section 3.2.3 and the task based approach in section 3.2.2 has desirable run-time properties. Both the APF approach in section 3.2.1 and the geodesic planning approach in section 3.2.4 requires solving an optimization problem (the spherical avoidance algorithm is not mentioned because it solves a much simpler optimization problem) and are therefore more suited to be part of a global planning or refinement layer. The task based avoidance scheme doesn't require any collision checking of the waypoints, but on the links of the USM directly and will therefore work in constant time. The therefore have the most suitable run-time properties. On the other hand, the lack of direct control on the motion of the head of the USM in contrast to the spherical avoidance approach weights negatively. Therefore, both these approaches are interesting to evaluate in practice to converge toward a final design. Such an analysis, however, is considered further work.

Kapittel 4

Control

4.1 Overall design

The control system design depends on the missions the USM have to execute. In Sverdrup-Thygeson et al. (2018) two main modes of operation are defined, respectively transport mode and work mode. The transport mode fulfills the purpose of moving the USM to a target position and the work mode of doing specialized tasks such as inspection and intervention. The two modes can thereby be characterized effectively by the restrictions they impose on the system. This chapter will present an overall control system architecture design for the transport mode, see figure 4.1, and how restrictions imposed by the definition of the mode itself and the navigation module, constrain it.

4.1.1 Transport mode

The transport mode presented in Sverdrup-Thygeson et al. (2018) face the following challenges during transit between two positions in some unstructured environment

- Obstacles may block the current path and real-time collision avoidance and re-planning is necessary.

- Moving through confined areas and narrow openings.

The USM should have to move over large distances, both in confined areas requiring high precision and maneuverability, and in large open areas defined by high speed and no requirements on maneuverability. The transport mode can therefore be characterized by two different operating conditions based on the required maneuverability and position accuracy imposed by the environment on the USM. Both these conditions require a guidance system that provides the desired course of the USM relative to the desired path, a dynamic control algorithm that compute the desired body forces and moments on the USM and a thruster allocation algorithm to actually generate forces to steer the USM. The modes differ in their use of kinematic tasks in the kinematic controller. The goal of the kinematic controller is to compute reference body and joint velocities, and doing so to fulfill a specific purpose, e.g. collision avoidance, path following or pointing the head toward features in the environment. Hoffmann (2018), Moe and Pettersen (2016) and Kohl et al. (2017) deploys a task based control scheme to solve the collision avoidance problem within the control system. The task based framework also allow the definition of a vision-based task, see e.g. Moe et al. (2016) for a practical example with a robot manipulator, but this may work at the expense of path following or collision avoidance performance. That is, the tasks have to be prioritized, which imply that lower priority tasks conflicting with higher priority tasks not necessarily gets executed.

Another use of kinematic tasks are for the sole purpose of generating reference velocities for path following, see e.g. Sverdrup-Thygeson et al. (2018) and Borlaug et al. (2018). Finally, Hoffmann (2018) use inverse kinematics to shape the USM while performing path following of straight line paths in 3D.

The constituent elements of the control system; the dynamic controller, the kinematic controller and the thruster allocation, have been investigated with good results. The focus in this chapter is therefore the overall approach to guidance and path following for curved paths in 3D, as most work is emphasized as straight line path following in previous work, see e.g. Borlaug et al. (2018), Hoffmann (2018), Wrzos-Kaminska (2018). The following questions will be addressed to converge towards an overall control system architecture for the transport mode while accounting for the aforementioned constraints;

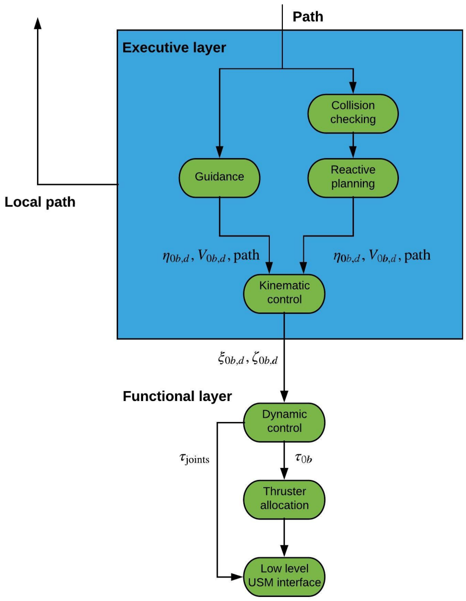
1. How can the guidance problem for curved paths be solved for USMs in 3D?
2. Can path following be solved in the classical sense, avoiding task-based collision avoidance?

This chapter is structured in the following way. Firstly, various choices for guidance controllers are discussed. Subsequently, a discussion on the kinematic control structure and how it interacts with the guidance system and the reactive planner will be presented. This is to provide a complete picture of the control system and to provide various choices before experimentation, presented in chapter 5.

Lastly, note that dynamic control design are excluded from this chapter as the intention is to solely construct a framework for USM guidance. The dynamical controller is assumed to take position and velocity references as input, $\xi_{0b,d}$, $\zeta_{0b,d}$ respectively as can be seen in figure 4.1, and track these with sufficient accuracy.

4.2 Line-of-sight (LOS) guidance

This section is dedicated to addressing the first question, *how to solve the guidance problem for curved paths in 3D*. Previously straight line 2D guidance techniques have been applied to USMs, inspired by Fossen (2011), in Sverdrup-Thygeson et al. (2018). A guidance strategy controlling heading were applied taking into account the cross-track error from the desired path. Kelasidi et al. (2014b) implemented a similar guidance law for USRs without thrusters and showed that convergence toward the straight line path can be guaranteed. Hoffmann (2018) implemented a straight line guidance technique in 3D computing the desired course angles in pitch and yaw based on the relative position between the USM and the path. This technique can be considered as a 3D extension of the 2D guidance algorithm presented in Fossen (2011). Wrzos-Kaminska (2018) implemented the same law as Hoffmann (2018) in addition to a geometric guidance law weighting directly the vertical and cross-track error. This section presents two different approaches to guidance.



Figur 4.1: Control system design transport mode.

The first technique is a guidance law based on the principles described in Moe et al. (2014), to ensure path convergence toward a Frenet frame following the curve. The results hold for general 3D curves satisfying the criteria in definition 2.5.4, but are restricted in the allowable desired pitch angle to avoid the singularity arising in the velocity transformation, equation (2.7), due to the Euler angle representation.

Remark 4.2.1. *Resolving the issues regarding the limited desired pitch angle can practically be avoided by adding further restrictions on the planned path relative to the USM, or by replacing the Euler angles with quaternions.*

The second technique is a geometric guidance law presented in Wrzos-Kaminska (2018) which provides a point of reference for the two former guidance laws. This section will describe these techniques and why either one of them should be chosen in the overall control system design.

4.2.1 3D guidance law

The 3D guidance law in Moe et al. (2014) makes the following assumptions

1. Roll is passively stabilized.
2. Ocean currents are constant, irrotational and bounded
3. The velocity is bounded by the available power.

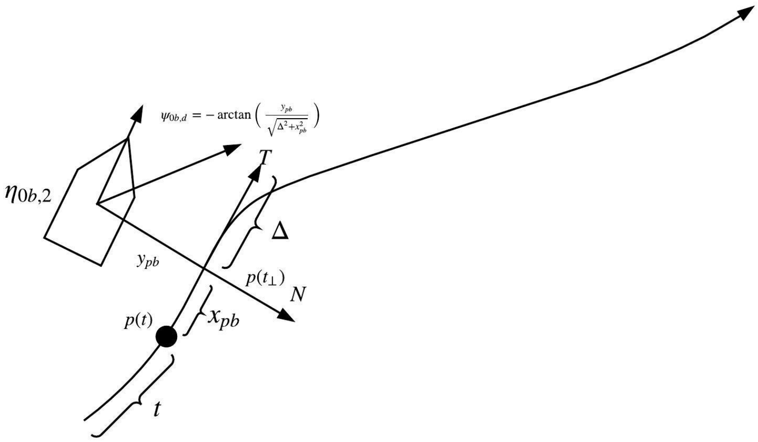
Remark 4.2.2. *Assumption 1 holds under the assumption that roll is stabilized by a lower level control loop.*

Consider the curve $p(\cdot) \in \mathbb{R}^3$ with a coordinate frame moving along it, called F_p, which implicitly describe the desired orientation. The guidance law described in Moe et al. (2014) suggest computing a course angle in pitch and yaw based on the geometric difference between the path and the pose of the robot. In particular, the progress along the path is iteratively updated. The relative position between the robot and the desired progress along the path then determine the course.

Figure 4.2 visualize how the guidance law works in 2D and provides a usual description of the relative geometry between the path and the robot. To drive the Frenet frame forward, an update law is defined as

$$\dot{t} = U_c \frac{\sqrt{\Delta^2 + y_{pb}^2}}{\sqrt{\Delta^2 + y_{pb}^2 + z_{pb}^2}} \frac{\sqrt{\Delta^2 + x_{pb}^2 + z_{pb}^2}}{\sqrt{\Delta^2 + x_{pb}^2 + y_{pb}^2 + z_{pb}^2}} + U_c \frac{x_{pf}}{\sqrt{\Delta^2 + x_{pf}^2 + y_{pf}^2 + z_{pf}^2}} \tag{4.1}$$

where $U_c = \sqrt{u^2 + v^2 + w^2}$ (respectively surge, sway and heave velocities) and Δ a look-ahead distance, under the assumption of no currents. Note that this update law assumes that the path is parametrized by arc length. The equations describing the desired reference angles are


$\psi_{0b,d} = -\arctan\left(\frac{y_{pb}}{\sqrt{\Delta^2+x_{pb}^2}}\right)$
T
$\eta_{0b,2}$
Δ
y_{pb}
$p(t_\perp)$
N
$p(t)$
x_{pb}
t

Figur 4.2: 2D conceptual image of the linearized guidance law. The intuition extends to 3D according to the equations (4.2) and extending the the Frenet frame with the unit binormal vector from equation definition 2.5.4

.

$$
\begin{aligned}
\chi_\theta &= \arctan\left(\frac{z_{pb}}{\sqrt{\Delta^2 + y_{pb}^2}}\right) \\
\chi_\psi &= -\arctan\left(\frac{y_{pb}}{\sqrt{\Delta^2 + x_{pb}^2 + z_{pb}^2}}\right)
\end{aligned}
\tag{4.2}
$$

Remark 4.2.3. *The look-ahead distance, Δ, scales the angles in (4.2) resulting in a less aggressive approach towards the path.*

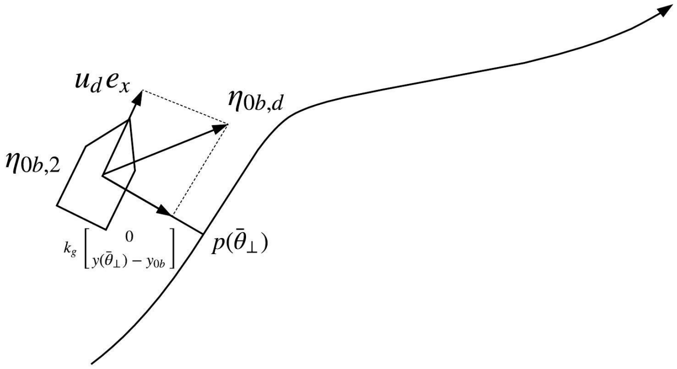
Remark 4.2.4. *Note that the current compensation used in Moe et al. (2014) are removed from equations (4.1) and (4.2).*

Equations (4.2) account for the fact that the timing law may be ahead or behind the current position. It is shown in Moe et al. (2014) that the guidance law can guarantee path convergence for an underwater vehicle, it is assumed that the method will work considering a single link of the USM. However, curved path following and path convergence can not be guaranteed for all the links. This would then have to be handled by the kinematic controller.

Finally, the direction for the reference velocity are consequently computed along the desired course angle. This can be represented by the deviation between the course angles (4.2) and the orientation of the USM base $\eta_{0b,2}$ in the base frame, F_b.

4.2.2 Geometric guidance

The concept of guiding the USM based on the point $p(t_\perp)$ can be formulated alternatively with a concept called geometric guidance, see Wrzos-Kaminska (2018). That is, the USM is simply guided in the direction of the point $p(t_\perp)$ toward the path by weighting the vertical and cross-track errors, see figure 4.3. The desired direction is then represented with a vector rather than with angles.


$u_d e_x$
$\eta_{0b,d}$
$\eta_{0b,2}$
$k_g \begin{bmatrix} 0 \\ y(\bar{\theta}_\perp) - y_{0b} \end{bmatrix}$
$p(\bar{\theta}_\perp)$

Figur 4.3: 2D conceptual image of the kinematic guidance law. The intuition extends trivially to 3D. Note that the error between the body and the path is defined in the body frame and not in the path frame as in section 4.2.1

$$\eta_d = u_d e_x - k_g \begin{bmatrix} 0 \\ y_{pb} \\ z_{pb} \end{bmatrix} \tag{4.3}$$

Remark 4.2.5. *k_g introduced in equation (4.3) is the reciprocal of Δ in equation (4.2). That is, a lower value results in a less aggressive approach toward the path and vice versa. To account for varying behaviours at different velocities, Wrzos-Kaminska (2018) scales it with u_d (the target velocity).*

Finally, note that this algorithm as well only guides the base link toward the path. Due to the lack of an update law, the guidance algorithm described in section 4.2.1 will be prioritized in simulations.

4.3 Task based control

The inverse kinematics are solved using the set-based control framework presented in section 2.3, inspired by Moe et al. (2016) and Chiaverini (1997). Moreover, this framework has also been applied earlier in the context of USMs, see e.g. Hoffmann (2018), Sverdrup-Thygeson et al. (2017). Hoffmann (2018) deployed two different sets of tasks; a guidance task during path following to bend the USM towards a desired path and a set of collision avoidance tasks to ensure that a number of links $m < n$ don't collide while traversing obstacles. Simulation results where provided to demonstrate the effectiveness of the two motion patterns separately. Sverdrup-Thygeson et al. (2017) provides a description of a motion control framework including a kinematic controller based on the SRMTP framework. The necessity of the kinematic controller in work mode is presented, but it is neglected with respect to the transport mode. Finally, Moe et al. (2016) provides a proof of the stability properties of set-based control deploying SRMTP. The conclusion states that the UGAS property are achievable provided that no tasks are conflicting. In that case, the lower priority tasks won't necessarily be completed. This section will present how the kinematic controller will integrate with the guidance layer to achieve path following and collision avoidance.

4.3.1 Input interface

As presented in figure 4.1, there are two modes of operation; a guidance mode and an interrupting collision avoidance mode. Both modes output ξ_{0b} and ζ_{0b} in addition to the desired path. This is to achieve the overall goal of path following while having the opportunity to quickly deliberate how to shape the body. Consider a goal specified by reference positions and velocities for the USM base as a main objective and ensuring that the rest of the USM follows the path as a secondary objective. To achieve this, the desired path has to be provided in addition to the desired base pose and velocity. If necessary, however, the USM should deviate from the path to avoid any body collision (this fail-safe are only considered conceptually). To summarize, this input specification provide the tools for a reactive kinematic layer to deliberate, independent of where the input comes from, to conclude on the safest choice for the USM.

4.3.2 Kinematic controller

The guidance laws from the previous section provide references for the base link of the USM which doesn't suffice to perform full body path following. That is, both end-effectors attached to the base have to be positioned along the path as well. The kinematic controller is used to ensure this objective with *end-effector tasks*. End-effector tasks can be defined simply as desired positions along the path relative to the target base link pose, see example 2.3.1. Note that these tasks propagate along the path at a fixed distance relative to the base link and are therefore, by construction, reachable. Results with this framework will be demonstrated by kinematic simulations in chapter 5.

Returning to the problem of task prioritization, as mentioned in the introduction of this chapter, consider the case when the head of the USM is supplied with sensors applied to navigation and mapping which should be controlled with care. Here a trade-off arise; How much should head control be prioritized over full body control? Should efficiency and convergence be sacrificed at the cost of smoother head movements? A good answer to these questions require taking into account how computer vision and SLAM interacts with the controller and are therefore out of the scope of this project. However, it is important to realize that there exists a trade-off, not due to path following against collision avoidance, but due to the necessity of orienting the navigational sensors *suitably* at all times. The tasks necessary to control this trade-off are presented next.

4.3.2.1 Avoiding body collisions

The success of 3D task based collision avoidance from Hoffmann (2018) illustrates that body collisions can be avoided in cases where the USM can't manage to follow the desired path appropriately. Then body tasks are defined for the necessary parts of the body, see section 3.2.2, to *push* the USM away from any nearby obstacles. These tasks will then be specified with high priority and the USM will enter an emergency mode. If these tasks are evoked, meaning that both the global path planner and the reactive planner is unable to compute a feasible path, then there are no additional fail-safe solutions to avoid collisions other than stopping the motion of the USM and potentially waiting for the global path planner to compute a new path around the new obstacle(s). Such a behaviour is fundamentally un-desirable as it entails a large deviation from the original plan. In accordance with the T-REX framework, see section 3.3, such deviations should be avoided, if possible, to reduce the dependency on time consuming higher level re-planning.

4.3.2.2 Path following task

The path following tasks consists of the end-effector tasks defined previously to ensure full body path following. Not much will be said about this task other than stating its necessity with regards to following the path provided by either a reactive or global planner. Under the assumption that the guidance algorithm ensures path convergence for the base, the end-effector tasks should be feasible. Under this assumption a prioritization between collision avoidance and path following can be deployed for curved paths. This provides a scaffold for the introduction of a final task, namely a field-of-view task.

4.3.2.3 Field-of-view task

Moe et al. (2016) provides an experiment including one equality task, a desired end-effector position, and two set-based tasks, namely a collision avoidance and a fov task. The fov task is defined as a desired spatial position that the end-effector should point towards.

Example 4.3.1. *Define* $\eta_{0b,2,d} \in \mathbb{R}^3$ *as a desired end-effector orientation. The fov task can thereby be defined by*

$$\begin{aligned} \sigma_{fov} &= \sqrt{(\eta_{0b,2,d} - \eta_{0b,2})^T(\eta_{0b,2,d} - \eta_{0b,2})} \\ \dot{\sigma}_{fov} &= \frac{(\eta_{0b,2,d} - \eta_{0b,2})^T}{\sigma_{fov}} J_{fov}(q)\dot{q} \end{aligned} \tag{4.4}$$

where J_{fov} *can be implemented as the geometric Jacobian choosing the dimensions involving the orientation.*

The importance of this task is the fact that the movement of the head can be controlled to point toward desired features in the environment. The experiments in Moe et al. (2016) includes collision avoidance as the highest prioritized task, the equality task as secondary, which would equal the path following task in this case, and the fov as a low-priority set-based task. This order makes intuitive sense, and to obtain the desired head behaviour the maximum joint velocity can be saturated. The implementation of this will resemble the presentation in Moe et al. (2016), but these details will not be pursued any further in this project.

Kapittel 5

Experimental design

This section will elaborate how experiments to test the control system are designed. In particular, experiments that can be carried out both in simulations and in practice to develop and validate performance are proposed. Several scenarios that target particular possible discrepancies in the control system are presented; straight lines, steady state turns and transient maneuvers. A set of performance variables are proposed for each module to summarize path following performance. Finally, simulation results for USM path following are presented in the context of this framework to conclude on which guidance controller to choose for the USM control system prototype.

5.1 Control experiments

To quantify the performance of path following approaches in the context of curved paths, or straight paths for that manner, a set of measures have to be defined. These measures play the role of response variables in an experimental context and are chosen to specify the performance of various aspects related to path following. Therefore, by definition, they are either geometric or internal to the USM. Measures of performance in the context of temporal requirements imply an additional set of tools, Rajan et al. (2000), and is considered further work because of the interactions with the mission manager and the global path planner. Absolute measures are in general also dependent on how the positions and velocities are estimated or measure (they can be considered as estimates either way, which guides the terminology used henceforth). Either way, there is an inherent uncertainty in the estimates which may or may not include drift. To maintain generality, no assumptions are made regarding how the estimates are obtained, and the measures are defined relative them.

Definition 5.1.1. Let $e^{h:j}_{\mathrm{var,i}}$ be the average deviation for variable(s) *var* for link or joint i between time steps h and j.

Example 5.1.1.

- *Full position link i*

$$e^{h:j}_{xyz,i} = \frac{1}{(j+1)-h} \sum_{k=h}^{j} ||\eta^{k}_{0i,1} - \eta^{k}_{0i,1,d}||_2 \ \in \mathbb{R} \tag{5.1}$$

- *Error joint i*

$$e^{h:j}_{q_i,i} = \frac{1}{(j+1)-h} \sum_{k=h}^{j} (q^{k}_{i} - q^{k}_{i,d})^2 \ \in \mathbb{R} \tag{5.2}$$

Remark 5.1.1. *Note that* $\eta^{k}_{0i,1}$ *in example 5.1.1 are estimates as discussed in the introduction of this section, abbreviated from the usual notation,* $\hat{\eta}^{k}_{0i,1}$.

In addition to the measures in definition 5.1.1, there are a lot of information in the time series response of the system variables. For the purpose of debugging and analysis, these should be stored as well. Finally, the following definition summarize a set performance measures for path following for USMs

Definition 5.1.2. The response variables classifying the path following performance are

- Convergence time until steady state
- Maximum overshoot
- Path alignment

- Joint deviation
- Size of control input
- Degree of chattering

Their respective definitions are provided in appendix A.

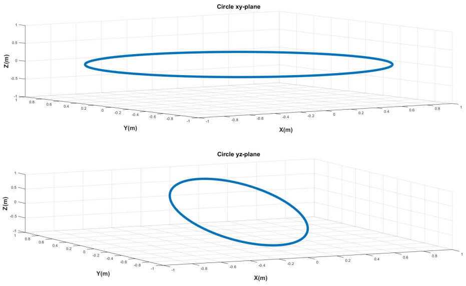
The rest of this section is dedicated to describing the importance of each of these performance measures in different experiments. Unfortunately, only experiments with appropriate results for the 3 former measures are can be provided. However, a discussion of the full range of measures in the context of the experiments are provided for documentation for later use.

5.1.1 Straight line path following

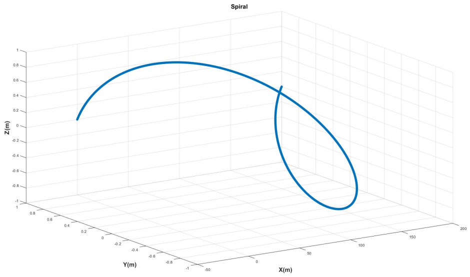
Consider the problem of following a straight line with an initial position deviating from the line. The goal is to guide the USM to the line and align its body along the path. Even though this is a conceivably straightforward test, it is useful for benchmarking the performance of the control system and for tuning as it is simple to replicate. With regards to the measures, it is particularly useful to use this test to quantify the overshoot resulting from the guidance algorithm, the convergence time and transient response of all the measures. Because the experimental setup emulates a operational space step input, these response variables are intuitive in comparison to applying a step response to a transfer function in classical control theory.

5.1.2 Steady state behaviour

Steady state turns can be represented by curves of constant curvature and torsion, namely circles, figure 5.1, and spirals, figure 5.2. Horizontal and vertical plane steady state can be tested with circles. This allows performance quantification in yaw and pitch separately provided that the roll is maintained constant at 0. Measures of particular importance for these tests are path alignment (both position and orientation), joint deviation and chattering. Because of the constant curvature/torsion, the path alignment and joint deviation should stay constant along with the size of the thruster control input. However, chattering may arise in such a setting as is interesting to measure. It is also interesting to investigate the difference in performance between pitch and yaw to verify whether are any anomalies in these measures. Finally, the curvature is varied to investigate how this scales the performance measures.



Figur 5.1: Circles in xy and yz planes respectively.



Figur 5.2: Spiral in 3D.

To test the interaction between pitch and yaw, a spiral of constant curvature and torsion is constructed, figure 5.2. Following the spiral implies interacting joint angles along the vertical and horizontal axis of the USM, a periodic rolling motion and, again, a constant thruster input.

Definition 5.1.3. The spiral is defined parametrized with distance along the global x-axis t as

$$S(t) = \begin{bmatrix} t & \cos(\frac{2\pi}{n}t) & \sin(\frac{2\pi}{n}t) \end{bmatrix} \tag{5.3}$$

where n scales the *frequency* of the oscillation of the spiral and thus the curvature and torsion of the curve.

Remark 5.1.2. *It can be shown that the curvature and torsion of the spiral in definition 5.1.3 are constant*

$$\begin{aligned} \kappa &= \frac{a^2}{1+a^2} \\ \tau &= \frac{a}{a^2+1} \end{aligned} \tag{5.4}$$

where $a = \frac{2\pi}{n}$. Note that these properties hold only from the perspective of the Serret-Frenet frame.

5.1.3 Curvature transient

When the USM have to perform a collision avoiding maneuver, it is likely that it have to deviate from straight line path following to follow a curved path. Curves of linearly, quadratic, etc. increasing and decreasing curvature could be considered depending on how critical the maneuver is. However, this is out of the scope of this work, as it isn't a necessary property to verify in a first prototype. How the USM control system is able to handle switching from straight lines to curves of constant curvature (e.g. an arc segment of a circle) can be considered a minimum viable test to establish how the USM will perform in collision avoidance scenario (this can be thought of as a step input in curvature). Such curves can for testing purposes be constructed with 2D Dubins curves for example (the 3D case will not be considered), see e.g. Fossen (2011).

The performance measures of interest are, by design, all transient responses and the time used to converge for a particular curvature transient. Results including transient testing are not presented.

5.2 Kinematic experiments

Simulation experiments are performed using a kinematic model of the USM. That is, to verify that curved path following is achievable under ideal conditions for the kinematic controller and guidance algorithms. That is, $\xi_{0b} = \xi_{0b,d}$ and $\zeta_{0b} = \zeta_{0b,d}$ are satisfied. This ideal assumption is made in Moe et al. (2016) regarding the joint angles for the kinematic controller. For the body position and velocity references, the assumption may not necessarily hold in practice due to the interaction between applied joint torques and thruster forces. Therefore, the subsequent results are presented for the purpose of verification and demonstration of the frameworks in sections 4.2, 4.3.2 and 5.1.

5.2.1 Straight line path following

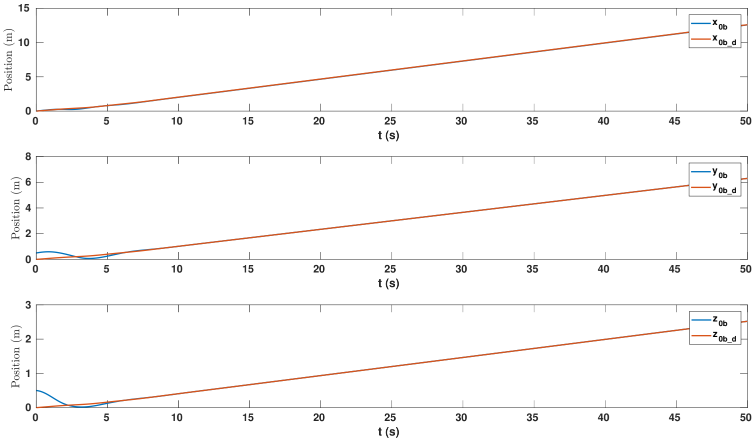
The straight line experiment demonstrates basic properties of the guidance and kinematic controllers with respect to convergence time and overshooting, see table 5.2.1. Note that the convergence in the x- and y-direction for the front is biased by the look-ahead distance which implies that its reference is ahead. This also affects the total convergence time. However, figure

5.2.2 Steady state behaviour

Results are shown in tables 5.2.2 and 5.2.2 for appropriate performance measures. Only parts of the circle are completed, see figure 5.10, to avoid redundant data. A notable observation is that the path alignment is worse for the smaller circle. As this experiment only involves a task for the rearmost link and the front end-effector, it can be seen that the rest of the body does not completely align with the smaller circle, figure 5.9. Figure 5.10 tell a partly different story when the curvature is decreased. It can additionally be seen from figures 5.7 and 5.8 that the convergence is slower for the smaller circle than the large circle. This may be an effect of geometry and the fact

Measures	xyz	x	y	z
SS base (m)	0.021441	0.025525	0.002746	0.0044042
SS front (m)	0	0	0.2122	0.048342
T base (s)	7.89	0.01	6.08	2.09
T front (s)	0	0	5.39	4.13
M base (m)	0	0.1533	0.21239	0.077775
M front (m)	0	2.5947	0.43913	0.35462
P (m)	0.021441	0.025525	0.21495	0.052747

Tabell 5.1: Results from steady state motion along a straight line for $T = 50s$. The notation follows from the definitions in appendix A. Zeros imply that the matrix entry is empty. $\epsilon = 0.01$ for the aft link and $\epsilon = 0.1$ for the front end-effector, and $n = 20$.



Figur 5.3: Rearmost link position against position reference along a straight line.

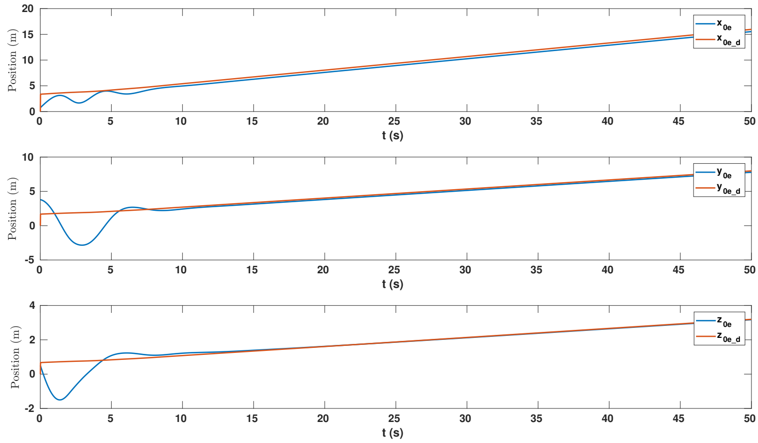
that the front end-effector has a longer way to travel to converge when the circle is smaller. Notably, the most prominent performance difference between the two circles are the difference in path alignment.

Measures	xyz	x	y	z
SS base (m)	0.013877	0.058738	0.017048	0.005322
SS front (m)	0.32522	0.59682	0.2696	0.048155
T base (s)	9.95	0.01	7.55	2.73
T front (s)	7.1	1.02	7.06	3.74
M base (m)	0	0.43685	0.40577	0.057395
M front (m)	0	0.83077	1.2073	0.40782
P (m)	0.3391	0.65556	0.28665	0.053477

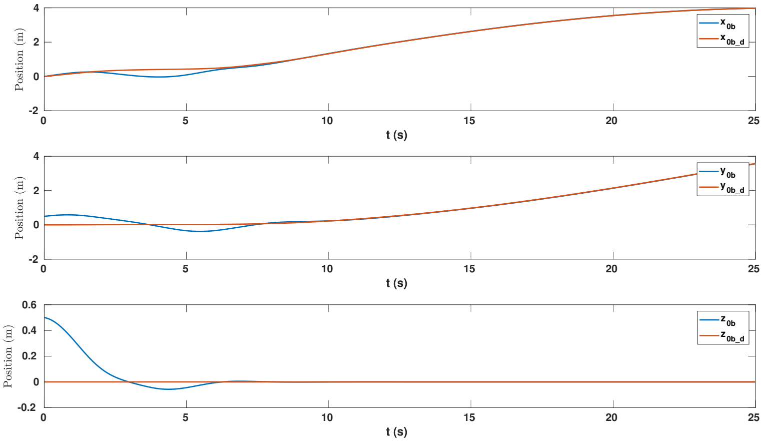
Tabell 5.2: Results from steady state motion around a circle of radii $R = 4m$ for $T = 25s$. The notation follows from the definitions in appendix A. $\epsilon = 0.01$ for the aft link and $\epsilon = 0.1$ for the front end-effector, and $n = 20$.

Measures	xyz	x	y	z
SS base (m)	0.0060305	0.032521	0.0082323	0.0024744
SS front (m)	0.18294	0.083143	0.16835	0.020861
T base (s)	10.26	0.01	7.65	2.75
T front (s)	8.95	6.15	8.94	3.76
M base (m)	0	0.45975	0.4229	0.056066
M front (m)	0	0.64601	1.0168	0.41386
P (m)	0.18897	0.11566	0.17658	0.023336

Tabell 5.3: Results from steady state motion around a circle of radii $R = 8m$ for $T = 50s$. The notation follows from the definitions in appendix A. $\epsilon = 0.01$ for the aft link and $\epsilon = 0.1$ for the front end-effector, and $n = 20$.



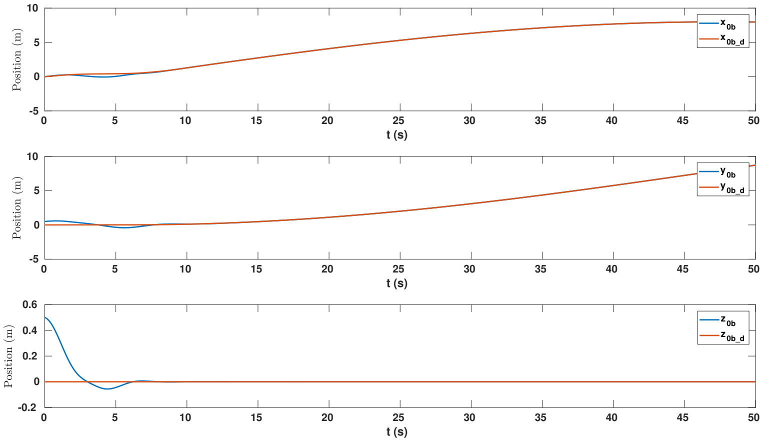
Figur 5.4: Front end-effector position against position reference along a straight line.



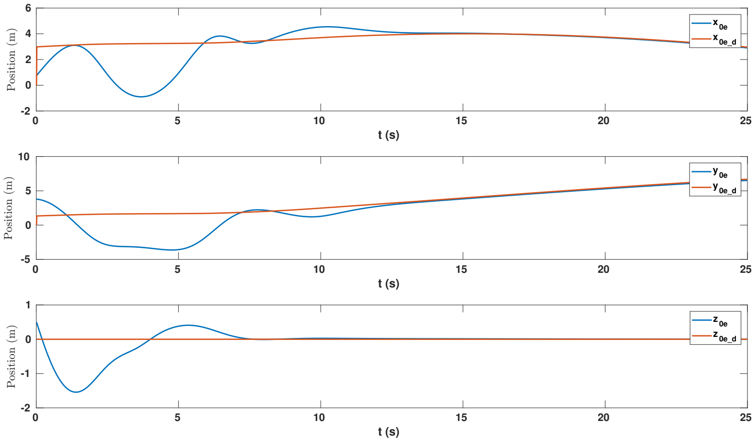
Figur 5.5: Rearmost link position against position reference around a reference circle with $R = 4$.

Measures	xyz	x	y	z
SS base (m)	0.0060305	0.032521	0.0082323	0.0024744
SS front (m)	0.18294	0.083143	0.16835	0.020861
T base (s)	10.26	0.01	7.65	2.75
T front (s)	8.95	6.15	8.94	3.76
M base (m)	0	0.45975	0.4229	0.056066
M front (m)	0	0.64601	1.0168	0.41386
P (m)	0.18897	0.11566	0.17658	0.023336

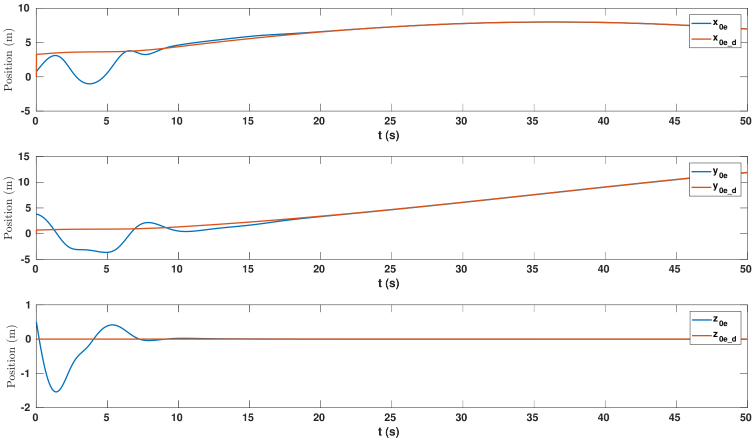
Tabell 5.4: Results from steady state motion around a circle of radii $R = 8m$ for $T = 50s$. The notation follows from the definitions in appendix A.



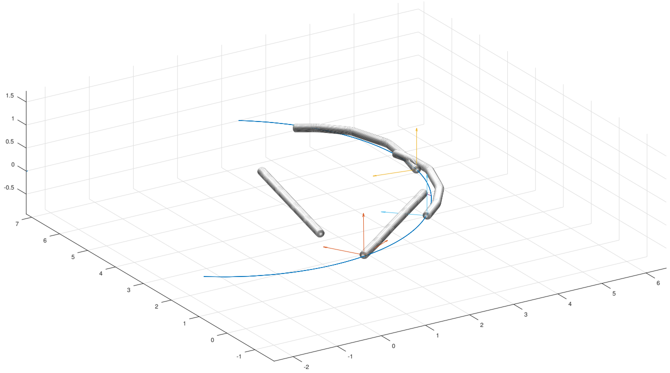
Figur 5.6: Rearmost link position against position reference around a reference circle with $R = 8$.



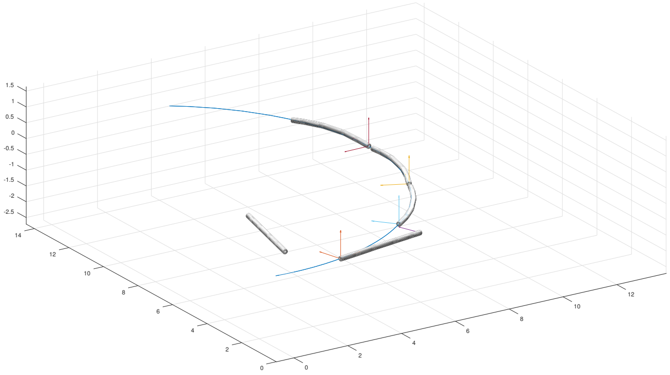
Figur 5.7: Front end-effector position against position reference around a reference circle with $R = 4$.



Figur 5.8: Front end-effector position against position reference around a reference circle with $R = 8$.



Figur 5.9: Snap shots of how the USM wraps around a circle of radii $R = 4m$. The coordinate system represents the base link aftwards and the blue line is a reference circle in the xy-plane.



Figur 5.10: Snap shots of how the USM wraps around a circle of radii $R = 8m$. The coordinate system represents the base link aftwards and the blue line is a reference circle in the xy-plane.

Kapittel 6

Conclusions and future work

Conclusion

This project have presented geometric criteria for path planning and collision avoidance algorithms for USMs, such that they can seamlessly integrate with the control system. A literary review of existing approaches to collision avoidance that can be applied to the USM were presented as well. Both the task based avoidance framework and the spherical avoidance algorithm shows promise. The methods are not mutual exclusive implying that both can be integrated into the deliberative architecture presented at the end of chapter 3.

A cascaded guidance and kinematic controller was proposed for full body path following. The correctness of the design was validated through kinematic simulations of the USM in MATLAB. In addition, the problem of task prioritization was addressed due to motion constraints imposed by navigational sensors such as cameras. The set of desired tasks; path following, collision avoidance and field-of-view, were defined. These tasks are to be implemented an tested in simulations and practice to verify their usefulness.

Finally, a framework for experimental design was tested to analyze path following performance for steady-state and transient maneuvers. The framework involves a tentative set of performance measures which will be used extensively during physical testing of the full control system in conjunction with the navigation and path planning modules. Simulation experiments was performed for steady-state testing two particular cases, straight line and a circle in the xy-plane, to demonstrate how to use the framework.

Further work

Further work include simulation testing of the full control system including all the kinematic tasks mentioned in section 4.3.2. Consequently, physical testing of the algorithms and verification of their performance in the proposed framework has to be done. Finally, integration with the navigation and path planning modules have to be tested to ensure that the desired assignments listed in section 3.3.1 work properly. The integration and testing of this full system will also include extending the framework from chapter 5 to include performance measures for computer vision, navigation, mapping and path planning. A part of this integration is completing the design of the deliberative architecture in section 3.3 and the full functionality required there. Lastly, note that the implementation of a collision avoidance algorithm such as the spherical avoidance approach from section 3.2.3 have low priority regarding the completion of the first prototype of the system.

Finally, note that the proposed further work are partly considered as collaborative work in a group, such as the extension of the experimental design framework and the completion of the deliberative architecture.

Tillegg A

Experimental measures

- **Time spent before convergence**
  Let k denote the discrete time instant k. Then the time spent before convergence can be defined as
  $$\mathrm{T} = \{k \in [0, N] :\ e_{xyz,b}^{\mathrm{k:k+n}} < \epsilon\} \tag{A.1}$$
  where N is the total number of discrete time steps in the experiment and $n \geq 1$ is a constant chosen to filter out transient.

- **Maximum overshoot**
  Let k satisfy (A.1) with $n = 1$, such that there exists no $0 < \bar{k} < k$ satisfying (A.1). Then the maximum overshoot can be defined as
  $$\mathrm{M} = \{p(t_{\bar{k}}) : e_{xyz,b}^{\bar{k}:\bar{k}} > e_{xyz,b}^{j:j} \forall j \in [k, N]\} \tag{A.2}$$
  where the notation $p(t_{\bar{k}})$ denotes the path evaluated at time instant $\bar{k}$ where $t_{\bar{k}}$ is the progress at this time.

- **Path alignment**
  Let k satisfy (A.1) with n chosen to ensure convergence. The path alignment can be obtained

$$\mathrm{P} = \sum_{i=1}^{n} e^{k:N}_{xyz,i} \tag{A.3}$$

or alternatively as an n-dimensional vector containing each of the errors

$$\mathbf{P} = \begin{bmatrix} e^{k:N}_{xyz,1} & \cdot\cdot & e^{k:N}_{xyz,n} \end{bmatrix} \tag{A.4}$$

The same can be defined for the orientation.

- **Joint deviation**
  Let k satisfy (A.1) with n chosen to ensure convergence. The joint deviation can be obtained

  $$\mathrm{J} = \sum_{i=1}^{n-1} e^{k:N}_{q_i,i} \tag{A.5}$$

  or alternatively as an (n-1)-dimensional vector containing each of the errors

  $$\mathbf{J} = \begin{bmatrix} e^{k:N}_{xyz,1} & \cdot\cdot & e^{k:N}_{xyz,(n-1)} \end{bmatrix} \tag{A.6}$$

- **Size of control input**
  Let $u_{\mathrm{thr},i}$ be the control input provided thruster i and n_* the number of thrusters, following the notation in Schmidt-Didlaukies et al. (2018). Define the vector of maximum control inputs

  $$\mathbf{U}_{\mathrm{max}} = \begin{bmatrix} \max u_{\mathrm{thr},1} & \cdot\cdot & \max u_{\mathrm{thr},n_*} \end{bmatrix} \tag{A.7}$$

- **Chattering**
  Let k satisfy (A.1) with n chosen to ensure convergence. Then define the time series of control inputs filtered with a simple moving average to remove chattering. The moving average is defined by $MA(\cdot)$, and the resulting time series become

$$\mathbf{U}_{\mathrm{MA}} = \begin{bmatrix} MA(u_{\mathrm{thr},1})^{k} & \cdot\cdot & MA(u_{\mathrm{thr},n_*})^{k} \\ MA(u_{\mathrm{thr},1})^{k+1} & \cdot\cdot & MA(u_{\mathrm{thr},n_*})^{k+1} \\ \cdot & \cdot\cdot & \cdot \\ \cdot & \cdot\cdot & \cdot \\ MA(u_{\mathrm{thr},1})^{N} & \cdot\cdot & MA(u_{\mathrm{thr},n_*})^{N} \end{bmatrix} \tag{A.8}$$

The chattering can now be defined as the variance of the control input around this moving average.

$$\mathbf{U}_{\mathrm{var}} = \frac{1}{N-k}\left[\textstyle\sum_{j=k}^{N}(MA(u_{\mathrm{thr},1})^{j} - u^{j}_{\mathrm{thr},1})^{2} \quad \cdot\cdot \quad \sum_{j=k}^{N}(MA(u_{\mathrm{thr},n_*})^{j} - (u_{\mathrm{thr},n_*})^{j})^{2}\right] \tag{A.9}$$

Remark A.0.1. *Note the abuse of notation of n in the context of path alignment and joint deviation. The n in the sum is the number of links.*

Remark A.0.2. *For transient analysis the maximum joint deviation and path alignment can be defined as*

$$\boldsymbol{I}_{max} = \left[\max e^{k:k}_{type,1} \quad \cdot\cdot \quad \max e^{k:k}_{type,n}\right] \tag{A.10}$$

with $n-1$ for joint deviation.

Bibliografi

Belkhouche, F. and Bendjilali, B. (2012). Reactive path planning for 3-d autonomous vehicles, *IEEE Transactions on Control Systems Technology* **20**(1): 249–256.

Borlaug, I.-L., Pettersen, K. and Gravdahl, J. (2018). Trajectory tracking for an articulated intervention auv using a super-twisting algorithm in 6 dof, *IFAC PapersOnLine* **51**(29): 311–316.

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.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley Sons, Ltd, Chichester, UK.

From, P. J. (2014). Vehicle-manipulator systems : Modeling for simulation, analysis, and control.

Hoffmann, B. H. (2018). *Path following and collision avoidance of USMs*, Ntnu.

Kelasidi, E., Pettersen, K. Y. and Gravdahl, J. T. (2014a). A waypoint guidance strategy for underwater snake robots.
URL: *http://hdl.handle.net/11250/2396241*

Kelasidi, E., Pettersen, K. Y. and Gravdahl, J. T. (2014b). A waypoint guidance strategy for underwater snake robots.
URL: *http://hdl.handle.net/11250/2396241*

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. Rob. Res.* **5**(1): 90–98.
URL: *http://dx.doi.org/10.1177/027836498600500106*

Kohl, A. M., Moe, S., Kelasidi, E., Pettersen, K. Y. and Gravdahl, J. T. (2017). Set-based path following and obstacle avoidance for underwater snake robots.
URL: *http://hdl.handle.net/11250/2481515*

LaValle, S. M. (2006). Planning algorithms.

Liljebäck, P., Pettersen, K. Y., Stavdahl, and Gravdahl, J. T. (2014). A 3d motion planning framework for snake robots.
URL: *http://hdl.handle.net/11250/286585*

McGann, C., Py, F., Rajan, K., Thomas, H., Henthorn, R. and McEwen, R. (2008). A deliberative architecture for auv control, *2008 IEEE International Conference on Robotics and Automation*, pp. 1049–1054.

Moe, S., Antonelli, G., Teel, A. R., Pettersen, K. Y. and Schrimpf, J. (2016). Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results, *Frontiers in Robotics and AI* **3**: 16.
URL: *https://www.frontiersin.org/article/10.3389/frobt.2016.00016*

Moe, S., Caharija, W., Pettersen, K. Y. and Schjolberg, I. (2014). Path following of underactuated marine surface vessels in the presence of unknown ocean currents, American Automatic Control Council, pp. 3856–3861.

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.

Rajan, K., Bernard, D. E., Dorais, G., Gamble, E. B., Kanefsky, B., Kurien, J., Millar, W., Muscettola, N., Nayak, P. P., Rouquette, N. F., Smith, B. D., Taylor, W. and Tung, Y.-W. (2000). Remote agent: An autonomous control system for the new millennium, *ECAI*.

Schmidt-Didlaukies, H. M., S, A. J. and Pettersen, K. Y. (2018). Modeling of articulated underwater robots for simulation and control, Trondheim, Norway.

Schøler, F. (2012). *3D Path Planning for Autonomous Aerial Vehicles in Constrained Spaces*, PhD thesis.

Siciliano, B., Sciavicco, L., Villani, L. and Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*, Advanced Textbooks in Control and Signal Processing, Springer London, London.

Sverdrup-Thygeson, J., Kelasidi, E., Pettersen, K. Y. and Gravdahl, J. T. (2018). The underwater swimming manipulatorx2014;a bioinspired solution for subsea operations, *Oceanic Engineering, IEEE Journal of* **43**(2): 402–417.

Sverdrup-Thygeson, J., Moe, S., Pettersen, K. Y. and Gravdahl, J. T. (2017). Kinematic singularity avoidance for robot manipulators using set-based manipulability tasks.
URL: *http://hdl.handle.net/11250/2463587*

Tapp, K. (2016). Differential geometry of curves and surfaces.

Wiig, M., Pettersen, K. and Krogstad, T. (2018). A 3d reactive collision avoidance algorithm for nonholonomic vehicles.

Wrzos-Kaminska, M. (2018). *Master thesis: Attitude control of USMs*, Ntnu.

Wynn, R. B., Huvenne, V. A., Bas, T. P. L., Murton, B. J., Connelly, D. P., Bett, B. J., Ruhl, H. A., Morris, K. J., Peakall, J., Parsons, D. R., Sumner, E. J., Darby, S. E., Dorrell, R. M. and Hunt, J. E. (2014). Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience, *Marine Geology* **352**: 451 – 468. 50th Anniversary Special Issue.
URL: *http://www.sciencedirect.com/science/article/pii/S0025322714000747*