Delft University of Technology

MSc Thesis
ME51010-20

A model predictive control approach for an autonomous underwater vehicle for quay wall inspection missions

Author:
Tijmen van Enckevort - 4552660

Abstract

The goal of this literature review was to derive which control method was most suited for the control of a robot that performs the mission of a quay wall inspection. Two types of robots, an Autonomous Tracked Vehicle (UTV) and an Autonomous Underwater Vehicle (AUV), were considered. Models were derived for both, which can be used in simulation tools that can model their working environment. These simulation tools showed that an AUV has more potential as a quay wall inspection robot. In order to provide a model for model-based controllers a control model is given. To determine the most suited control method for the robot the classical control methods Bang-Bang control and Proportional Integral Derivative (PID) control are compared with the model-based control methods of Linear Quadratic Regulator (LQR) control and Model Predictive Control (MPC). MPC has proven to be the best suited control method due to its ability to handle Multi-Input Multi-Output (MIMO) system like the AUV, handle constraints and compared to the other control methods it has a better performance. A Lyapunov-Based Model Predictive Control (LMPC) method is a variant on MPC that is more robust and stable, which can improve the control of the AUV even more.

Contents

1 Introduction

Over 70% of the world is covered in water, and over the past ages humans have had the tendency to build their cities and harbors around that water to exploit it in terms of resources, logistics and recreational facilities. These exploits require a lot of infrastructure. This infrastructure, for example harbors and canals, must not fail and break down, else catastrophic harm will be done to the surroundings, see Figure 1. Often quay walls are part of the water infrastructure. Quaywalls are large, partly submerged structures in the form of a wall that keeps the water in its desired location, and not on the other side of the quaywall. Quaywalls are vulnerable for cracks and other faults that may not always be visible from above the water line. This requires the need for continuous inspection. At the moment this is largely done with sophisticated diving teams. Teams consisting of multiple divers, an operational facilitator and a safety inspector are inspecting quaywalls meter by meter to look for any faults that can be disastrous to the structural integrity of the wall. This is a tough and time-consuming job, and requires lots of expensive equipment [1].

Figure 1: The collapsed Grimburgwal in Amsterdam [2]

In order to improve this process and make it more efficient it is proposed to build an autonomous underwater vehicle (AUV). This AUV should be able to autonomously follow a path along the quaywall, and take measurements to detect any faults the wall might have. Measurements can be in the form of taking photographs or videos which can be turned into a 3D model of the quaywall with the use of a technique called photogrammetry [3][4][5]. This 3D model can be used to detect faults in the wall, and compare 3D models of quaywalls of previous years to see if cracks are propagating.

A robot functions properly when it can sense its environment and act upon it in a correct manner. In order to do this the robot needs to fulfill the 'see-think-act' cycle, see Figure 2 [6]. The 'see-think-act' cycle is a set of phases that a robot has to continuously run through. It starts in the real world environment, where the robot is operating. It then has to perceive this environment and extract the correct information from this, for example its own location within the environment (the 'see' phase). From here it can plan its next action in the path planning phase, for example a new location for the robot to drive to or an interaction with a gripper and an object (the 'think' phase). This planned action then has to be acted upon in the motion control phase, by a control algorithm that sends inputs to the robot actuators (the 'act' phase). Afterwards, the robot has acted upon its environment, and the cycle starts over again.

The last step in this cycle, the 'act' phase, incorporates the low-level control of the robot. This step involves a desired reference path from its path planning phase. A control algorithm is used to follow that reference trajectory. There are a lot of possible control algorithms possible that can each follow the reference trajectory. This

literature survey aims to answer the research question as to what control method is most suited for the control of a robot that performs the mission of a quay wall inspection, and how this control algorithm can be designed and implemented.

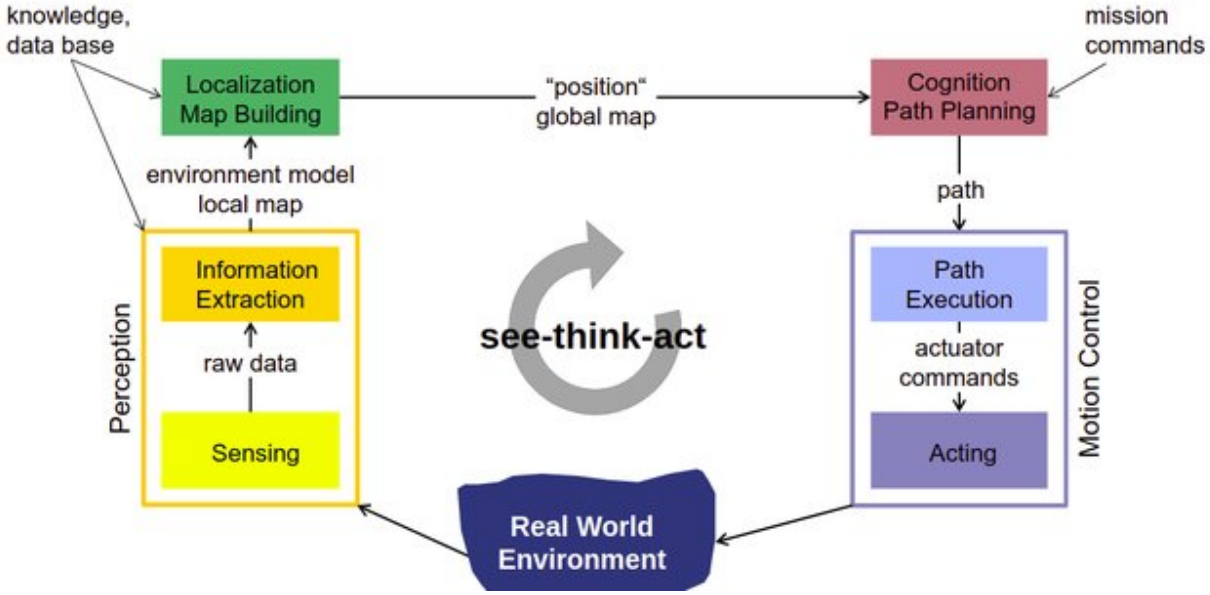


Figure 2: See-Think-Act cycle for autonomous robots [6]

The problem that arises from this is twofold. First, simulation tools of the robot has to be reviewed, as this allows for easier and quicker testing and validation of the AUV. Besides, the robot has yet to be built, thus the simulation is also a necessity to be able to test the control algorithm. This simulation should ideally consist of all the steps of the 'see-think-act' cycle and resemble the true working environment of the robot as much as possible.

Next to this the implementation of the control algorithm should be reviewed. This involves the research to what control algorithm is best suited, computing a model of the robot, dealing with constraints of the robot, designing a method to reject disturbances to the robot and tuning the control algorithm.

1.1 Scope of the inspection mission

As the simulation environment should resemble the real working environment of the robot it is important to determine the scope of the working environment of the robot. The robot should operate along two types of quay walls. It has to operate along flat quay walls made out of bricks, see Figure 1. Next to this it should also be able to inspect quaywalls made out of hexagonal steel structures, so called sheet pile walls, see Figure 3. This makes it that it should be able to operate in a 2D environment, as well as a 3D environment with potential obstacles in the way of the robot. Ideally the robot is able to navigate around the obstacles in either it's path planning or low-level control stage.

Figure 3: Sheet pile wall [7]

The waters the robot will operate in has negligible waves and underwater currents, in other words, the water will be calm. The water can be murky, so sensor information from for example visual sensors will be noisy, or only accurate up to a certain distance.

1.2 Report layout

This literature review continues with a the modelling of the UTV and AUV in Section 2. Here the kinematic and kinetic models are explained that are needed to describe the robot mathematically. In Section 3 the tools that are needed to simulate the underwater robots are explained. These are used to test the robot, and also test its control algorithm. Next, the model that can be used in the controller are explained in Section 4. This controller model is a linearized model of the simulation model, and is used in the model-based controllers. Possible types of control algorithms, which can be used to control the robot, are explained in Section 5. Both classical control methods as well as model based controllers are explained here . Section 6 contains a conclusion on the results and answers the research question. Finally, Section 7 presents the further planning of the master thesis.

2 Robot Model

There are numerous robots or vehicles that operate under the water surface existing already. They are used for pipeline inspections, the manipulation of valves or other levers on underwater structures, or the laying of pipelines for example. Roughly, their working principles can be partitioned into two categories.

First, there are vehicles that move or drive over the floor of the body of water [8] [9] [10] [11]. They operate on this 2D space and often use tracks to actuate themselves. These underwater tracked vehicles (UTV) are used in applications like drilling and taking samples of the sea floor, cleaning swimming pools or hulls of ships, or the digging of trenches for pipelines. Depending on their use their gravitational pull is larger than their bouyancy, meaning they sink. These types of UTV's are often used on the seabed where they need to interact with their environment. Other types are neutrally bouyant, meaning they do not sink nor rise in the water. These types often use a thruster to create a force orthogonal to their driving surface with which they can adhere themselves to the driving surface. They are often used in the cleaning of swimming pools or hulls of ships where they also need to be able to drive on surfaces that are not oriented horizontally.

Secondly, there are robots that are swimming or floating freely in the body of water [12] [13] [14]. These autonomous underwater vehicles (AUV) or remotely operated vehicles (ROV) operate in the 3D space of body of water. They use fins, rudders or thrusters to actuate themselves. They can be used on very long inspection missions where they map the sea floor, be used to interact with underwater structures and for example turn valves, or be used in the place of divers on short missions where the main goal is inspection of a structure. Vehicles that are used in long inspection missions are often very slender with a low hydrodynamic drag. They use fins and a small number of thrusters to actuate themselves. This causes their energy usage to actuate themselves to be low, but this does come at the cost of poor mobility and maneuverability. AUV's that are used to inspect or interact with structures often have a poorer performance regarding hydrodynamic drag. They consist of a frame with a multitude of thrusters mounted onto it. This causes them to have a high mobility and maneuverability, but they cannot autonomously sustain long missions.

In order to converge to the best possible design of the inspection robot this literature review continues with the modelling, simulation and control of the neutrally buoyant UTV and the AUV with thrusters. They are assumed to be suited the best for the inspection mission, and can form the basis of the robot that has yet to be designed and built.

2.1 Autonomous Tracked Vehicle

This section explains the models for both the UTV and the AUV that are needed to describe the robots in a mathematical manner. These models can be used in the design, simulation, motion planning and control phases of the robot and are thus essential in the development of the underwater inspection robot.

2.1.1 Introduction ATV

Underwater tracked vehicles (UTV) are used in numerous applications under the water surface. They can be used to clean the hull of ships, do inspection missions on the seabed floor or dig trenches to lay pipelines. An UTV contains two parallel tracks with which it can drive around on the floor of the body of water. Depending on the type of robot and its application it is possible to make the robot neutrally bouyant so that it floats freely in the water. This neutral bouyancy can be combined with a thrusters on the topside of the body of the robot. This thruster creates a force orthogonal to the surface plane the robot is driving on. Using this force and the neutral buoyancy this creates the possibility for the robot to be driven on planes that are in any orientation. Thus, with this it is able to drive on the surface of quay walls, and inspect it. An example of a robot that uses a thruster on the topside of its body to let it drive around on walls can be seen in Figure 4. An UTV can be equipped with a camera, or a rig of camera's, and lights for illuminating the seabed. With these sensors it is able to inspect a quay wall. An UTV is also able to drive over small obstacles, like the fouling on the hull of a ship, or small rocks on the bottom of the sea floor.

Figure 4: Keelcrab robot with control tablet. [15]

2.1.2 Extended Kinematic Model with slip

In order to be able to simulate and control the robot a model of the robot is needed that describes the real world physics of the robot onto a mathematical description. The kinematic model describes the motion of the body of the robot without considering the forces and moments that are causing or resulting from those motions. A simple kinematic model of a tracked vehicle robot would be a differential drive model. However, various studies have proven this model is not accurately modelling the robot motion. This is mainly due to track slippage, where the track slips over the ground surface. Slip is defined as the difference between the velocity measured by a track (theoretical velocity) and the actual forward vehicle velocity (real linear velocity). It is defined as [16]:

$$i_j = 1 - \frac{v}{\rho \omega_j} \tag{1}$$

Here, i_j is the slip of track j, v is the forward vehicle velocity, ρ is the track radius and ω_j is the angular velocity of track j.

There are two types of slip at play in the tracked vehicle robot. First there is lateral slip. The lateral slip is caused by the centrifugal force which occurs due to the tracked vehicle turning at high velocities. According to [16] it can be neglected for tracked vehicles moving at low velocities. The second type of slip is the longitudinal slip. The main cause of longitudinal slip are the track-soil interactions. These track-soil interactions are complex and hard to model and for that reason it is preferred to empirically estimate the slip i_j for different soil types the robot is driving on.

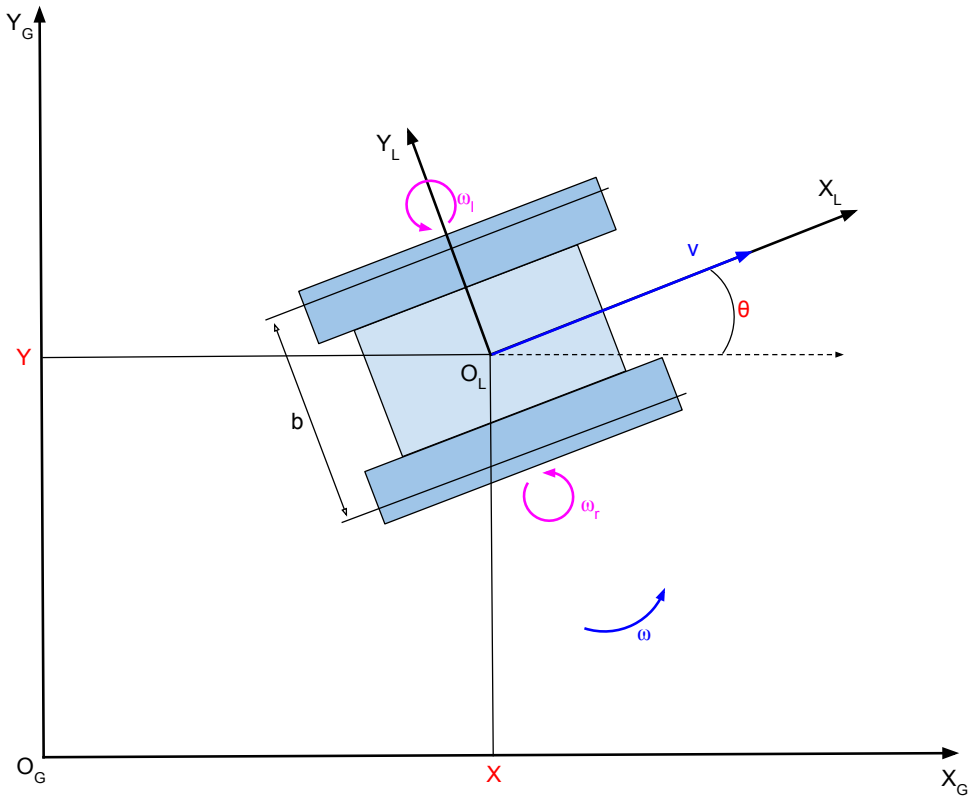


Figure 5: Motion of a tracked mobile robot. ($O_gX_GY_g$ is the reference or inertial frame and $O_LX_LY_L$ is the local robot frame.) [16]

The position and orientation is represented by $[x\ y\ \omega]^T \in \mathbb{R}^3$ without any slip. In absence of slip the linear velocities of the tracks are given by:

$$\begin{aligned} v_r(t) &= \rho\omega_r(t) \\ v_l(t) &= \rho\omega_l(t) \end{aligned} \tag{2}$$

Here, $v_r \in \mathbb{R}$ and $v_r \in \mathbb{R}$ are the linear track velocities of the right and left tracks, and $t \in \mathbb{R}^+$ is the continuous time.

When modelling the robot in slip conditions the track velocities change to:

$$\begin{aligned} v_r^{sl}(t) &= v_r(t)(1-i_r)(t)) = \rho\omega_r(t)(1-i_r(t)) \\ v_l^{sl}(t) &= v_l(t)(1-i_l)(t)) = \rho\omega_l(t)(1-i_l(t)) \end{aligned} \tag{3}$$

Using this, the extended kinematic model of the robot with slip conditions is given by:

$$\begin{bmatrix} \dot{x}^{sl}(t) \\ \dot{y}^{sl}(t) \\ \dot{\theta}^{sl}(t) \end{bmatrix} = \begin{bmatrix} \frac{v_r^{sl}(t)+v_l^{sl}(t)}{2} cos\theta^{sl}(t) \\ \frac{v_r^{sl}(t)+v_l^{sl}(t)}{2} sin\theta^{sl}(t) \\ \frac{v_r^{sl}(t)+v_l^{sl}(t)}{b} \end{bmatrix} \tag{4}$$

Where the position and orientation of the robot is represented by $[x^{sl}\ y^{sl}\ \omega^{sl}]^T \in \mathbb{R}^3$. This can also be seen in Figure 5.

2.2 Autonomous Underwater Vehicle

2.2.1 Introduction AUV

In order to perform an underwater quay wall inspection mission it is also possible to use an autonomous underwater vehicle. This is a robot that can autonomously operate below the water surface. It does so by floating or swimming in the 3D space that is bound by the water surface, the quay walls and the floor of the body of water. For actuation it can make use of thrusters. The inspection can be done with the use of a rig of camera's, and some lights to help visualize the quay wall in the poor lighting conditions that are present below the water surface. An example of a possible setup would look like Figure 6.

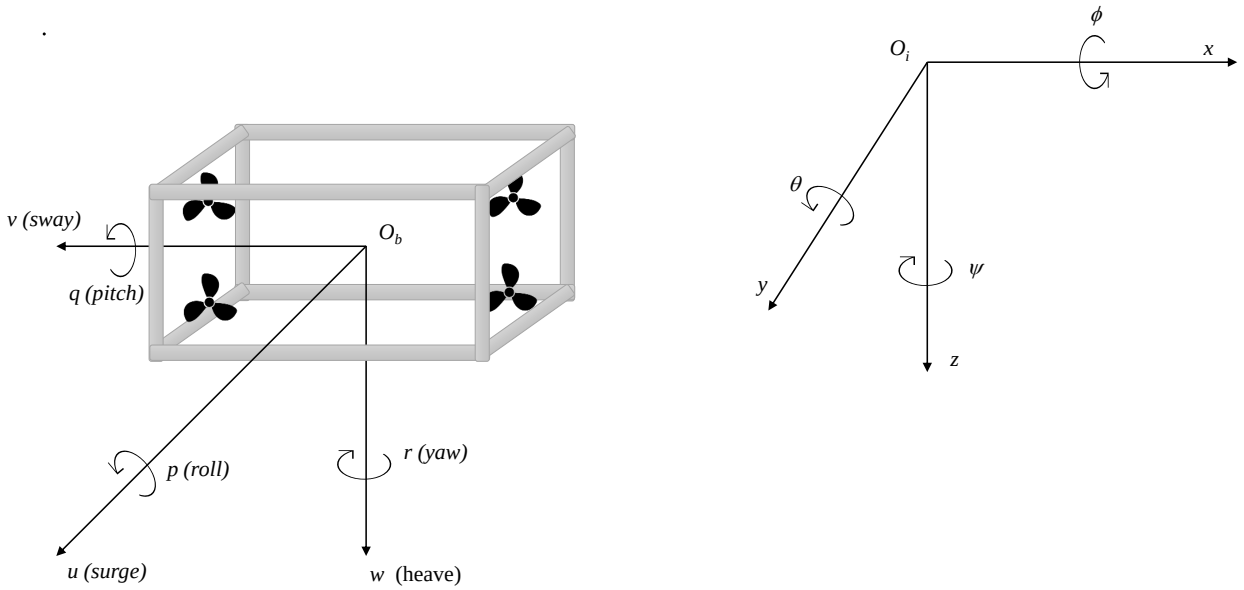


Figure 6: Motion of an underwater robot. O_i is the reference or inertial frame NED and O_b is the local robot frame BODY.

The underwater world that the robot operates in is a 3D environment with 6 Degrees of Freedom (DOF), which can be seen in Figure 6. For consistency this paper uses the SNAME notations to express the physical states of the robot, which can also be found in Table 1 [17]. Note that the position and orientation of the robot is expressed in the North East Down (NED) inertial frame of the robot whereas the velocities, and forces and moments of the robot are expressed in the Body frame of the robot.

Degree of Freedom	Position and orientation in Euler angles in NED frame	Linear and angular velocities in Body frame	Forces and moments in Body frame
surge	x	u	X
sway	y	v	Y
heave	z	w	Z
roll	ϕ	p	K
pitch	θ	q	M
yaw	ψ	r	N

Table 1: SNAME notation

In order to express and control the robot different models can be used. To express the robot a simulation model can be used that incorporates kinetics and kinematics. It is extensive and incorporates as much of the forces and moments that are acting on the body as possible. The control model is generally less extensive, and captures the main dynamics in order to control the robot in a sufficient manner. This chapter further describes

the simulation model, where the kinematic model is described in Section 2.2.2, and the kinetic model is described in Section 2.2.3.

2.2.2 Kinematic Model AUV

The kinematic model describes the motion of the body of the robot without considering the forces and moments that are causing or resulting from those motions, see also Figure 6. The kinematics in Euler angles that describe the motion of the underwater vehicle can be found in Equation 5 [17].

$$\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\Phi}} \end{bmatrix} = \mathbf{J_e}(\phi) \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{R_b^i}(\boldsymbol{\Phi}) & \mathbf{0_{3x3}} \\ \mathbf{0_{3x3}} & \mathbf{T_b^i}(\boldsymbol{\Phi}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{bmatrix} \tag{5}$$

Here:

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3, \;\; \boldsymbol{\Phi} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in S^3, \;\; \boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3, \;\; \boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3 \tag{6}$$

$\mathbf{R_b^i}(\boldsymbol{\Phi})$ is the linear velocity transformation matrix from the body frame BODY to the inertial frame NED given by:

$$\mathbf{R_b^i}(\boldsymbol{\Phi}) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\theta s\phi \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi s\theta c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{7}$$

Where $s = sin, c = cos$, and $t = tan$.

$\mathbf{T_b^i}(\boldsymbol{\Phi})$ is the angular velocity transformation matrix from the body frame BODY to the inertial frame NED given by:

$$\mathbf{T_b^i}(\boldsymbol{\Phi}) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \tag{8}$$

Note that this kinematic model is given in the Euler representation and only holds when $\theta \neq 90$. If this does not hold a quaternion representation can be used instead.

2.2.3 Kinetic Model AUV

In order to describe the motion of the robot body whilst taking into account the forces and moments that are acting on the body a kinematic model of the robot can be used. The equations of motion (EOM) of a kinetic robot model are given in Equation 9. Here, six different parts can be identified which together describe the motion of the robot. These six parts are further explained in the following sections.

$$\underbrace{\mathbf{M_{RB}}\dot{\mathbf{v}} + \mathbf{C_{RB}}(\mathbf{v})\mathbf{v}}_{Rigid-Body} + \underbrace{\mathbf{M_A}\dot{\mathbf{v_r}} + \mathbf{C_A}(\mathbf{v_r})\mathbf{v_r} + \mathbf{D}(\mathbf{v_r})\mathbf{v_r}}_{hydrodynamics} + \underbrace{\mathbf{b}(\mathbf{R_b^i})}_{restoring} = \underbrace{\boldsymbol{\tau_{thrusters}}}_{actuation} + \underbrace{\boldsymbol{\tau_{cable}}}_{umbilical} + \underbrace{\boldsymbol{\tau_{dist}}}_{disturbances} \tag{9}$$

2.2.3.1 Rigid Body Motions

The rigid body kinetics of an underwater robot can be described by [17]:

$$\mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{RB} \tag{10}$$

Here, the $\boldsymbol{\nu}$ is the generalized velocity vector in the body frame, where $\boldsymbol{\nu} = [u, v, w, p, q, r]^T$ and the $\mathbf{M}_{RB}$ is the rigid body mass matrix. It is defined as:

$$\mathbf{M}_{RB} = \begin{bmatrix} m\mathbf{I}_{3\times3} & -m\mathbf{S}(\mathbf{r}_g^b) \\ m\mathbf{S}(\mathbf{r}_g^b) & \mathbf{I}_b \end{bmatrix} = \begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -Ixz \\ mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zx} & I_z \end{bmatrix} \tag{11}$$

$\mathbf{S}(\mathbf{r}_g^b)$ is a skew-symmetric matrix, where $\mathbf{r}_g^b$ is the location of the Centre of Gravity (CoG) with respect to the center of origin. It is defined as:

$$\mathbf{S}(\mathbf{r}_g^b) = \begin{bmatrix} 0 & -z_g & y_g \\ z_g & 0 & -x_g \\ -y_g & x_g & 0 \end{bmatrix} \tag{12}$$

Note that when the CoG is the same as the centre of origin: $\mathbf{S}(\mathbf{r}_g^b) = \mathbf{0}_{3\times3}$ and

$$\mathbf{M}_{RB} = \begin{bmatrix} m\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_b \end{bmatrix} \tag{13}$$

$\mathbf{C}_{RB}(\boldsymbol{\nu})$ is the Coriolis and centripetal matrix. If it is noted that $\boldsymbol{\nu}_1 := \boldsymbol{\nu}_{b/n}^b = [u, v, w]^T$, and $\boldsymbol{\nu}_2 := \boldsymbol{\omega}_{b/n}^b = [p, q, r]^T$, $\mathbf{C}_{RB}(\nu)$ can be written in the following lagrangian parametrization:

$$\mathbf{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} \mathbf{0_{3\times3}} & -m\mathbf{S}(\boldsymbol{\nu_1}) - m\mathbf{S}(\mathbf{S}(\boldsymbol{\nu_2})\mathbf{r_g^b}) \\ -m\mathbf{S}(\boldsymbol{\nu_1}) - m\mathbf{S}(\mathbf{r_g^b})\mathbf{S}(\boldsymbol{\nu_2}) & -\mathbf{S}(\mathbf{I_b}\boldsymbol{\nu_2}) \end{bmatrix} \tag{14}$$

2.2.3.2 Hydrodynamics

To accurately model the forces of the surrounding fluid on the body moving under the water surface one has to solve the Navier-Stokes equations. However, these are computationally expensive and are not feasible in a real-time setting. Another approach is the use of a so-called added mass matrix. This added mass matrix can be seen as a virtual mass added to the inertia of the body that is caused by the accelerating of the surrounding fluid due to the motion of the body.

The added mass matrix is defined as:

$$\mathbf{M}_A = \begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \tag{15}$$

The hydrodynamic Coriolis and centripetal matrix is defined as:

$$\mathbf{C}_A = \begin{bmatrix} \mathbf{0}_{3\times3} & -\mathbf{S}(\mathbf{A}_{11}\boldsymbol{\nu}_1 + \mathbf{A}_{12}\boldsymbol{\nu}_2) \\ -\mathbf{S}(\mathbf{A}_{11}\boldsymbol{\nu}_1 + \mathbf{A}_{12}\boldsymbol{\nu}_2) & -\mathbf{S}(\mathbf{A}_{21}\boldsymbol{\nu}_1 + \mathbf{A}_{22}\boldsymbol{\nu}_2) \end{bmatrix} \tag{16}$$

Also, the relative velocity $\mathbf{v_r}$, which is caused by water currents, is given by $\mathbf{v_r} = \mathbf{v} - \mathbf{v_c}$, where $\mathbf{v_c} = [u_c, v_c, w_c, 0, 0, 0]$

The coefficients of the added mass matrix $\mathbf{M_A}$ can be found numerically or empirically.

2.2.3.3 Damping

Besides the added mass to the body, there is also a damping effect due to the body moving in a fluid. The damping effects are highly complex and not fully understood and are thus hard to model. An approach to model the damping is by defining:

$$\mathbf{D}(\boldsymbol{\nu}_r) = \mathbf{D_l} + \mathbf{D}_{nl}(\boldsymbol{\nu}_r) \tag{17}$$

It is composed of the linear damping matrix $\mathbf{D_l}$ and the nonlinear damping matrix $\mathbf{D}_{nl}(\boldsymbol{\nu}_r)$. In the assumption that the symmetric robot moves at low velocities and below the water surface the linear damping matrix is due to viscous skin friction, whereas the nonlinear damping matrix is due to vortex shedding (and maybe quadratic damping and higher-order terms?). [18].

$$\begin{aligned} \mathbf{D_l} &= diag(X_u, Y_v, Z_w, K_p, M_q, N_r) \\ \mathbf{D_{nl}} &= diag(X_{u|u|}, Y_{v|v|}, Z_{w|w|}, K_{p|p|}, M_{q|q|}, N_{r|r|}) \end{aligned} \tag{18}$$

The coefficients of the nonlinear damping matrix $\mathbf{D}_{nl}(\boldsymbol{\nu}_r)$ can be described by one of the terms of the Morison equation [18]:

$$\mathbf{F}_D = \frac{1}{2}\rho C_D A_p |u| u \tag{19}$$

Here, ρ is the water density, C_D is the drag coefficient, A_p is the projected cross-sectional area and u is the velocity of the robot body. The drag coefficients C_D are to be determined empirically, as are the coefficients of the linear damping matrix $\mathbf{D}_{nl}$.

2.2.3.4 Restoring forces

The hydrostatic or restoring forces acting on the robot are caused by both gravitational forces, and buoyancy forces due to the robot being submerged in water. The gravitational force is defined as $W = mg$, where it acts on the CoG of the robot in the positive direction along the heave axis.

The buoyancy force is the force is caused by the displacement of water due to the robot being submerged under water. It is defined as $B = \rho \nabla g$ where it acts on the Centre of Buoyancy (CoB). This is the volumetric centre of the body. It acts in the negative direction along the heave axis. Both gravitational and buoyancy forces are expressed in the inertial frame i as:

$$\mathbf{f}_G^i = W \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \mathbf{f}_B^i = B \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{20}$$

In the body frame b the restoring forces and moments are given by:

$$\mathbf{b}(\mathbf{R}_b^i) = \begin{bmatrix} {\mathbf{R}_b^i}^T(\mathbf{f}_G^i + \mathbf{f}_B^i) \\ \mathbf{r}_G^b \times \mathbf{R}_b^i \mathbf{f}_G^i + \mathbf{r}_B^b \times {\mathbf{R}_b^i}^T \mathbf{f}_B^i \end{bmatrix} \tag{21}$$

These are the two forces acting on the robot in hydrostatic conditions, i.e. when there is no relative flow of water across the robot body ($\mathbf{v_r} = 0$).

2.2.3.5 Actuation

The body of the robot is being actuated with underwater thrusters. For a fully actuated system the amount of independently orientated thrusters r should be equal to the number of DoF n the robot is planning to move in. In the case that $r > n$ the system is over actuated and the thruster inputs have to be allocated using an optimization algorithm. One way to model the individual thruster output is:

$$f = ku \tag{22}$$

Here, f is the thruster force, k is the thruster coefficient, and u is the input. The thruster coefficient can be determined from the relationship between the thruster force and the rotations per minute of the thruster.

The sum of the force and moments from the thrusters $\boldsymbol{\tau_{thrusters}}$ are expressed as:

$$\boldsymbol{\tau}_{thrusters} = \mathbf{Tf} \tag{23}$$
$$= \mathbf{TKu} \tag{24}$$

Here, $\mathbf{T}$ is the Thruster Allocation Matrix, $\mathbf{f} = [f_1, f_2, ..., f_n]$ is the thruster force vector with n thrusters, $\mathbf{K}$ is a matrix with thruster coefficients, and $\mathbf{u}$ is the vector with thruster inputs.

2.2.3.6 Umbilical Forces

This section is about the forces that are present due to the tether or umbilical cord that is coming from the 'mothership' to the AUV. This umbilical cord can exert forces acting on its attachment point on the robot. These forces are caused by drag due to currents/robot vehicle velocity in the tangential direction of the umbilical cord, and hydrostatic forces due to gravity or buoyancy.

According to [13] the umbilical forces can be modelled as:

$$\tau_{cable} = \begin{bmatrix} -\frac{1}{4}\rho d C_d \int_0^h |u_r| u_r \mathrm{d}z \\ -\frac{1}{4}\rho d C_d \int_0^h |v_r| v_r \mathrm{d}z \\ 1.2 w_{cable} h \\ r_z \tau_{cable}(1) \\ r_z \tau_{cable}(2) \\ 0 \end{bmatrix} \tag{25}$$

Here, the ρ is the density of the water, d is the diameter of the cable, C_d is the drag coefficient of the cable, h is the depth of the AUV relative to the 'mothership', w_{cable} is the weight of the cable submerged in water, and u_r and v_r are the velocities in the u and v direction of the current relative to the umbilical cord. The u_r and v_r can be caused by both the water current velocity as the velocity of the AUV in the horizontal plane.

The AUV will follow a zig zag pattern where the longest direction of travel will be in the vertical z direction. This makes that the u_r and v_r will generally be small which causes that the umbilical forces in the X and Y direction of the AUV can be neglected as they are scaling quadratically with their corresponding velocity directions. Furthermore it can be assumed the umbilical is neutrally bouyant, with its center of gravity and center of buoyancy being in the same position. This causes the underwater weight of the cord w_{cable} to be approximately equal to 0. Taking also in account that the AUV is not expected to operate at very deep depths, which keeps the length h of the umbilical cord relatively small. This diminishes all the elements in the umbilical force vector even further. Following these arguments it can be argued that the umbilical forces acting on the AUV can be assumed to be negligible.

$$\tau_{cable} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{26}$$

2.3 Discussion

Both the UTV and AUV can be modelled according to the models described in this Section. As the UTV is operating in a 2D space and can be accurately modelled using its kinematics only this model is a lot simpler.

The Fossen model of AUV is a lot more complex, and involves both kinetics and kinematics. In both cases the parameters of the model have to be determined numerically or experimentally. These models are the mathematical representation of the robots, but do not explain or describe the characteristics of the robots intuitively. In order to to this the models can be used in simulation tools to simulate the robot into an operating environment. This will create a more intuitive mission scenario and will help in the design of the robot and its controller.

3 Simulation Tools

In order to properly design and build a robot it is necessary to test it a lot. Physical tests with real robot components are not really suited for this, especially at the early stages of the design process. This is because the parts can be expensive, it is easy to make a mistake and break a part, ordering and swapping parts and setting up the testing environment can be cumbersome and costs a lot of time. For that reason, especially in the early stages of the design process, it is often preferred to simulate the robot in a simulated environment on a computer. Here, the parts are ready to be used and performing test missions of the robot can be done rapidly. This speeds up the design process, whilst also making it a lot less expensive.

It is preferred for the simulation environment and physics engine to resemble the true real life conditions as closely as possible. If this is the case the robot can be accurately modelled and the best design choices can be made whilst preventing any design mistakes from being made. This ultimately leads to the best designed robot and accompanying algorithms.

Gazebo classic is an open-source robot simulator with a robust physics engine. It is able to seamlessly work together with the Robot Operating System (ROS). In Gazebo the robot and all its parts can be simulated inside a realistic environment. Besides this there are plugins available for specific robot kinematics and kinetics, for example the tracked vehicle plugin and Plankton, which can be used for the simulation of the tracked robot and the underwater robot respectively. An example of the Gazebo interface with an underwater robot can be seen in Figure 7.

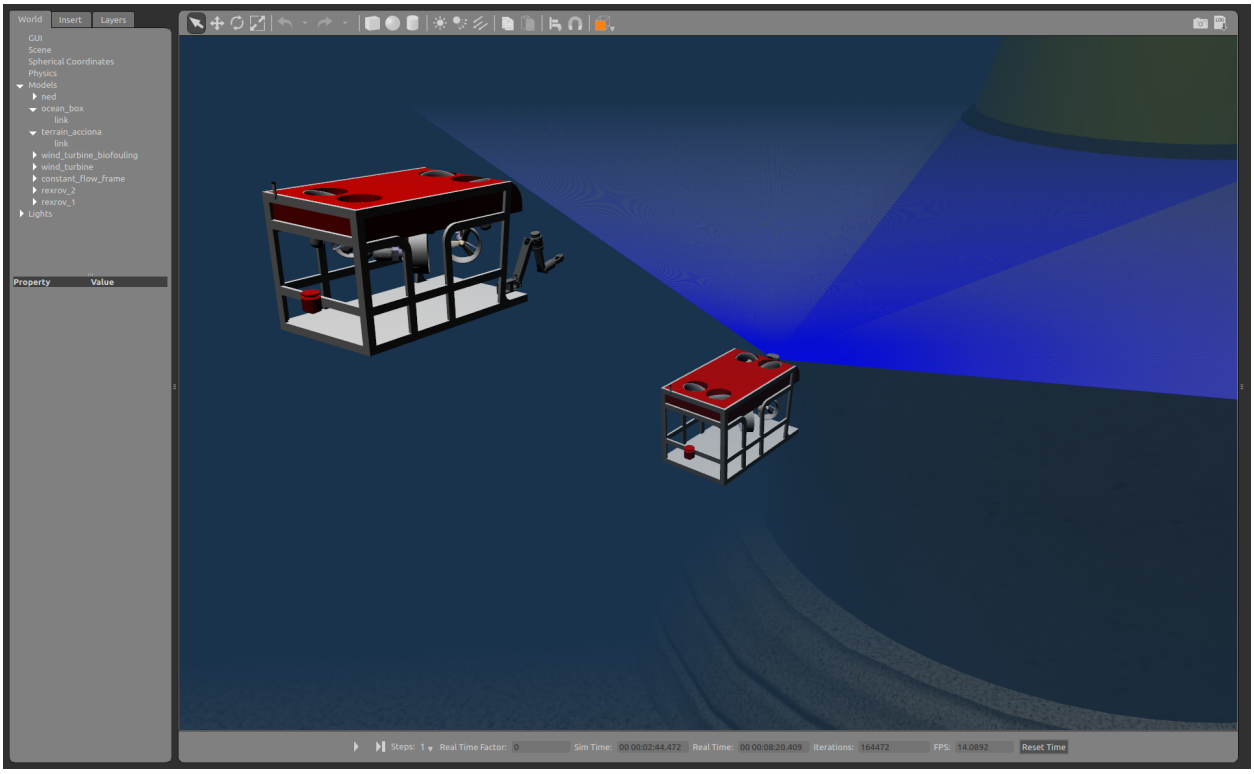


Figure 7: Gazebo simulator with underwater robot [19]

The Robot Operating System is software that can be used to control and instruct robots. It is widely used in both academia and industry and is the operating system of choice for lots of robots. Currently there are two versions of ROS available: ROS and ROS2. As expected, ROS2 is the successor of ROS. ROS and ROS2 are open source and mostly developed by volunteers or companies that are willing to contribute to the ROS community. Both versions of ROS use a lot of standalone nodes that can communicate with each other in order to control the robot. This architecture makes that it is easy to add, remove or amend nodes to your robot depending on the specific needs.

UTV

3.1 Tracked vehicle plugin

The default version of the Gazebo simulator does not natively support tracks, and for this reason a plugin has to be used that can simulate robots with tracks. The tracked vehicle plugin is a plugin that can simulate non-deformable tracks in a way that is computationally inexpensive and physically plausible [20]. This plugin first computes the Instantaneous Centre of Rotation (ICR) of the tracked vehicle. For a differential drive robot this ICR is in the orthogonal direction to the robot wheels. However, for a tracked vehicle this ICR is dependent on the slip of the tracks, and is not necessarily orthogonal to the robot wheels.

Next, it uses the effect of a so-called contact surface motion (CSM). This CSM first changes the friction direction to be tangent to the ICR instead of parallel to the tracks of the robot. With makes sure the robot is actually able to make turn, instead of only having a forward velocity. Then, it sets the relative velocity of the tracks to a nonzero velocity so that the robot moves forward, and computes the friction forces based on that. This workaround is not physically accurate, as one would preferably set the forces that act on the tracks, which in turn change the velocity of the tracked robot. However, [20] has shown that it is the most physically plausible and accurate way of simulation the tracked vehicle in real-time compared to other tracked vehicle simulation methods. For this reason it is the preferred method to be used for simulation of the tracked vehicle concept. Besides, it is already readily available as a plugin for the Gazebo simulator, which eliminates the need to build a new plugin based on another method.

AUV

3.2 Underwater Simulation

In order to accurately model and simulate the effects of the underwater environment on the robot a plugin on top of the Gazebo physics engine has to be used. One of such plugins is Plankton. Plankton is a plugin that is based on the plugin uuv simulator. However, uuv simulator is only updated till the Melodic version of ROS, and consequently needs to be used on a 18.04 distribution of the Linux operating system. In the current project setting this is not preferred as there is preference on the 20.04 distribution of Linux, together with the new version of ROS 2. The Plankton plugin is based on the uuv simulator, with much of the same structure and algorithms, but it works with ROS 2 instead of ROS Melodic. This makes it the preferred plugin for the underwater robot.

The Plankton plugin adds the hydrodynamic and restoring effects to the rigid body dynamics that already exist in Gazebo, see also Equation 9. Furthermore, it is able to model the thrusters according to the model from Equation 24. Using this, the full equations of motion according to [17] and 9 can be simulated in the simulation environment. This simulation environment includes a visual layer of the water surface, and by using 3D models of the quay walls or other underwater structures a simulation environment can be built that is also visually accurate to the real operating environment. This is important as the robot will use cameras for its navigation and inspection, and thus, besides a physically accurate simulation environment, a visually accurate simulation environment is needed as well.

There are also some control algorithms present that are used by the example robots in the plugin. One of those is a cascaded PID algorithm, which uses several PID control layers 'cascaded' on top of each other. In this case there is a PID controller on the position and orientation, a PID controller on the linear and angular velocities, and a PID controller on the linear and angular accelerations of the robot.

Besides the simulation of the physics of the robot there are also several tools available in the plugin to aid in the design of the robot. There are scripts that can calculate the thruster allocation matrix $\mathbf{T}$, example robots that have a working simulation with the correct added mass and damping matrices, and templates for the thrusters and other robot parts like the base frame and the sensors.

3.3 Preliminary conclusion

During the project a choice has to be made between the two concepts, as fully designing and building both robots would be too time extensive and costly. To aid in the making of this choice two preliminary simulations of the robots using the above plugins were made. These simulations are not fully accurate, for example, they might not have the correct slip coefficients, or the correct added mass coefficients to be physically correct. However, they can be used to quickly see any major drawbacks or flaws to the design, which can not be solved in any other way.

The simulation of the UTV showed two major flaws. First, the robot is only to operate on a 2D surface. Here it is able to drive over obstacles, but only minor ones, as larger obstacles lead to the robot being tipped over. This is a feasible solution for any quay wall that is relatively smooth without any obstacles. However, it is preferred to also be able to inspect quay walls that do not have a smooth and flat surface, and to also be able to cross obstacles like underwater ladders. Secondly, the robot tracks will loosen and perturb the fouling that is present on the quay walls. This in turn makes that the water surrounding the robot will be murky with lots of particles in it. This is not preferred as this will severely block the vision of the cameras, and the cameras are critical as they are needed for navigation and inspection.

A preliminary simulation of an AUV showed a better performance. As it is able to operate in 6 degrees of freedom it is possible to avoid any obstacles that might be present, as well as being able to inspect quay walls that are not smooth or flat. Also, as it is not touching the quay wall and thus not interfering with its fouling the creation of murky water is less present. As the robot is operating in a 6 DOF environment and besides kinematics needs to take into account the kinetics as well the model is a more complex. Furthermore, it is not straightforward to determine all the coefficients in the EoM of the robot. This makes the simulation and controller more complex to design, which can slow down the design and building process of the robot.

After considering both preliminary simulations it was determined to continue with the AUV instead of the UTV. This was mainly decided due to its ability to avoid obstacles that might be present on the quay wall. The coming chapters of this report will outline the controller architecture of this robot.

The simulation models of the robots might not be suited in the model-based controller described in Section 5. This is because the simulation models tend to be computationally expensive, which is not desired in real-time controllers. The model-based controllers can make use of a simpler version of the simulation model which is less computationally expensive. This is explained in Section 4.

4 Control Model

This section discusses the components necessary to control an underwater vehicle, such as the model used in the controller, the error or reference trajectory, the constraints of the underwater vehicle and the disturbances present in the environment. This control model can be used in the model-based controllers in Section 5. If the true state of the robot can be determined via the simulation software in simulation environment or a motion capture system in a test environment, the observer model can be omitted as it is redundant.

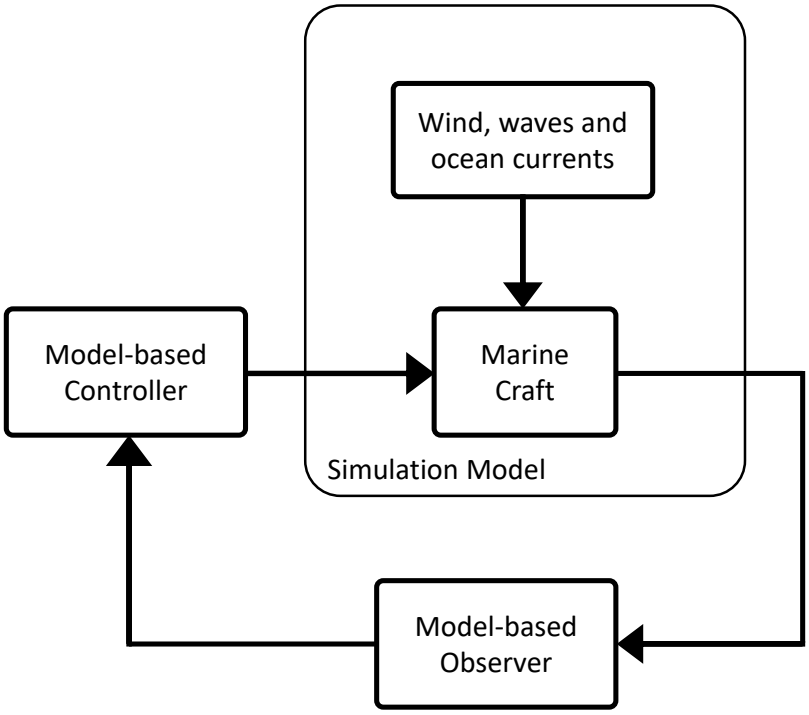


Figure 8: Different models used in underwater vehicles [17]

4.1 Linear State Space Model

The full 6 DoF equations of motion 9 might not be suited for the control algorithm as they can be too computationally expensive for use in real-time applications. For this reason [17] states that a second, more simplified version of the simulation model is preferred, see also Figure 8. This control model does not model all the dynamics and environmental effects and makes some assumptions. The control model can then be used in the real time control of the underwater vehicle.

In the assumption of a neutrally buoyant and highly metacentric stable underwater vehicle in low-speed applications the EoM 9 can be linearized. That is, $\phi, \theta \approx 0$, and $\boldsymbol{\nu} \approx \mathbf{0}$, which means the EoM 9 can be linearized about $\phi = \theta = 0$, and $\boldsymbol{\nu} = \mathbf{0}$. This leads to $\mathbf{C}(\mathbf{0}) = \mathbf{0}$, $\mathbf{D}_n(\mathbf{0}) = \mathbf{0}$, and $\mathbf{g}_0 = \mathbf{0}$ which gives:

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \underbrace{\mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\mathbf{0}} + \underbrace{[\mathbf{D} + \mathbf{D}_n(\boldsymbol{\nu})]\boldsymbol{\nu}}_{\mathbf{D}_\nu} + \underbrace{\mathbf{g}(\boldsymbol{\eta})}_{\boldsymbol{G}\boldsymbol{\eta}_p} + \underbrace{\mathbf{g}_o}_{\mathbf{0}} = \boldsymbol{\tau} \tag{27}$$

Where:

$$M = M_{RB} + M_A \tag{28}$$
$$C(\boldsymbol{\nu}) = C_{RB}(\boldsymbol{\nu}) + C_A(\boldsymbol{\nu}) \tag{29}$$

This gives:

$$\dot{\boldsymbol{\eta}}_p = \boldsymbol{\nu} \tag{30}$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}\boldsymbol{\nu} + \mathbf{G}\boldsymbol{\eta}_p = \boldsymbol{\tau} \tag{31}$$

These equations of motion can be written in the linear time-invariant (LTI) state-space model:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{32}$$

$$\tag{33}$$

Where:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M^{-1}G} & -\mathbf{M^{-1}D} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{M^{-1}} \end{bmatrix}, \quad \mathbf{x} = [\boldsymbol{\eta}_p^\top, \boldsymbol{\nu}^\top]^\top, \quad \mathbf{u} = \boldsymbol{\tau} \tag{34}$$

For a neutrally buoyant underwater vehicle, $\mathbf{G}$ is given by:

$$\mathbf{G} = diag(0, 0, 0, (z_g - z_b)W, (z_g - z_b)W, 0) \tag{35}$$

This linearized model is valid for highly metacentric underwater vehicles in low speed applications and as it is not computationally expensive it is suited as a model for the control algorithm.

4.2 Constraints

The system of the underwater vehicle can have both input and output constraints. Both of which should be fulfilled in order for the system to operate properly and not fail.

The input constraints are the constraints on the inputs of the system. The inputs of the system are the input currents given to each of the thrusters. These inputs are bound, as the thruster can only handle a maximum amount of current before it fails. These bounds are given by the minimum (reverse) and maximum (forward) inputs possible to the thruster. As the force outputs of the thruster scales linearly with the thruster input 22 this also implies a minimum and maximum bound on the force output of the thrusters.

The output constraints are the constraints on the outputs of the system. The outputs of the system are the position and orientation of the underwater vehicle. The output constraints are given by the distance to any obstacles or quay walls. The most critical constraint will be the distance to the quay wall which is in the positive x surge direction, and the bottom of the floor of the body of water, which is in the positive z heave direction. Situations where the robot moves above the surface of the water, or moves away from the quay wall are not critical and should not be taken into account in the control algorithm. The output constraints should be hard constraints, as the robot should not touch either the wall or the floor as this can severely damage the robot.

Besides output constraints on the location and orientation of the robot it is also possible to include output constraints on the rate of change of the location and orientation of the robot, see **source needed**. These output constraints will prohibit the underwater vehicle from sudden movements and make the overall motion trajectory of the underwater vehicle more smooth and less jittery. This can be a desirable trait when making photographs of the quay wall as sudden movements can cause blurry photos.

4.3 External Disturbances

According to [17] there are three main causes of disturbances to marine crafts due to environmental effects. These causes are ocean currents, waves, and wind effects.

The ocean current is the subsurface flow of water with a velocity $\boldsymbol{\nu}_c$. This causes an effect to the hydrodynamics of the underwater vehicle, expressed in the relative velocity of the water $\boldsymbol{\nu}_r$ along the underwater vehicle. The relative velocity is given by $\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c$ and $\boldsymbol{\nu}_c = [u_c,\ v_c,\ w_c,\ 0,\ 0,\ 0]^\top$. If $\boldsymbol{\nu}_r$ can be measured or approximated it can be included into the equations of motion of the underwater vehicle in the following way:

$$\mathbf{M_{RB}}\dot{\boldsymbol{\nu}} + \mathbf{C_{RB}}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{M_A}\dot{\boldsymbol{\nu}}_r + \mathbf{C_A}(\boldsymbol{\nu_r})\boldsymbol{\nu_r} + \mathbf{D}(\boldsymbol{\nu_r})\boldsymbol{\nu_r} + \mathbf{b}(\mathbf{R_b^i}) = \boldsymbol{\tau} \tag{36}$$

If waves are present in the environment they can also have an effect on the underwater vehicle. Although the vehicle is fully submerged, close to the water surface waves will still have a effect on the bodies below it, see [21]. However, as these waves are complex to measure and model, and the application areas of the robot generally do not have significant waves an attempt to model these waves can be excluded from the control model.

Wind can also have an effect on marine crafts. However, in situations where the marine craft is submerged under the water surface continuously these effects can be neglected **source**.

4.4 Discussion

By using a linear model of the system instead of a more accurate nonlinear model there is an introduction of inaccuracies of the model when moving far away from the linearization state. This is not preferred, as the robot will then not behave properly as expected. This trade-off, between the computational efficiency of the linear model and the accuracy of the nonlinear model has to be balanced in practical tests. It could be that the computers used are powerful enough to calculate the nonlinear model in realtime applications of the controller, or that the inaccuracies of the nonlinear model are negligible.

The assumption that waves do not have an effect on the robot model might not be entirely accurate when the environmental conditions the robot has to be used in prove to have larger waves than anticipated. Take for example an inspection of quay walls at the beginning of a harbor, close to sea. In this case the assumption does not hold anymore. This can be solved by only performing an inspection mission when there are no considerably waves present and the water is calm.

Due to inaccuracies in the observed state of the robot the output constraints might not hold all the time. If constraints are set on a distance to the quay wall, but the true state of the robot is not accurately known, it can cause issues in the sense that the robot can collide with the wall while the controller thinks it does not. This problem can be solved by knowing the true state in the simulation environment, but in real world applications this might prove to be more difficult.

As the control models by themselves do not yet provide the necessary control actions to drive the AUV towards its desired reference state a controller has to be designed. There are multiple options regarding controller designs which will be discussed in the following Section 5.

5 Motion Control

This section is about the motion control of an underwater vehicle. In order for an underwater vehicle to follow a trajectory or reference path a controller has to be implemented that can give the correct inputs to the thrusters of the underwater vehicle. This section is divided into two parts. The first part is on classical control techniques, where bang-bang control and PID control are discussed, and the second part is on model-based controllers, where LQR and model predictive control are discussed.

5.1 Classical Control

Classical control is a subset of control designs that solves dynamical systems in the laplace domain. They have been used extensively in underwater vehicles, where the often preferred method is that of PID control. These controllers use the error of the plant state compared to the reference state to compute the actuator inputs of the system

5.1.1 Bang-Bang control

A bang-bang controller [22][23] is a type of controller that switches its output suddenly based on the input of the controller. It is a feedback controller, where the input to the controller is the error of the plant state compared to the reference state. An error margin is added to the error. If the error is outside of these error margins the controller suddenly switches from output state. A bang-bang controller only has two outputs, and is also called an on-off controller. This switching behaviour can lead to time-optimal behaviour, in the sense that a plant can reach (but not necessarily sustain) a reference state in the shortest amount of time if the actuators of the plant are acting on their minimum or maximum bounds. The controller is straightforward to implement and understand, which make it a popular controller for simple and low-cost systems.

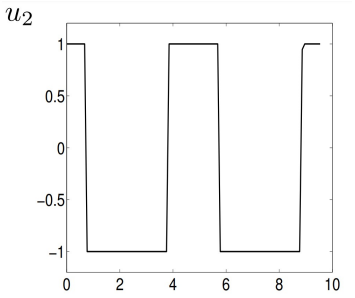


Figure 9: Example output of bang-bang [22]

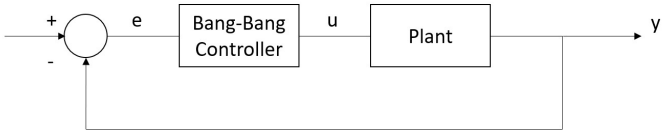


Figure 10: A bang-bang controller scheme

In underwater vehicles bang-bang controllers are used in several configurations. In [23] an AUV uses a PID controller for path following, and when it reaches the final stage of a docking manoeuvre it switches to a bang-bang controller for its final alignment. Due to the bang-bang controller being more aggressive the AUV was able to dock successfully. In [24] two different types of controllers are used for the pitch angle, depth, and roll angle of an AUV. For the depth of the AUV a bang-bang controller is implemented.

Because of its sudden switching of the outputs into the maximum or minimum states of the actuators of the plant this controller requires a lot of effort of the actuators of the plant. Because of this the controller is undesired for some systems, as it creates a lot of wear and tear on the actuators. This is also the case for underwater vehicles, where it generally is undesired to switch between the maximum and mimimum states of the thrusters suddenly and frequently. In order to diminish this it is possible to apply fuzzy logic controller to the underwater vehicle, as demonstrated by [25]. This fuzzy logic controller creates a more gradual transition from minimum to maximum actuation, which is beneficial to the wear and tear of the thrusters. Another option for underwater vehicles is the use of a piecewise-constant (PWC) control, which also aims to decrease the number of switchings of the actuator [26]. It is able to achieve this with only a small deviation on time and state error compared to a bang-bang controller, but with significantly less actuator switchings.

5.1.2 PID control

A PID controller gives an input to a system based on the error of that system state compared to the reference state. This input is build up from an proportional (P), integral (I), and derivative (D) gain. The proportional gain scales linearly with the error. This gain provides an input to the system whenever there is an error between the reference and the system state, see also Figure 11. The integral term is a gain that scales with the integral of the error. This gain gets larger the longer a certain error persists, and by doing so makes that the steady state error converges to zero. With this integral term the steady state disturbances that are present on the system can be rejected. The derivative term is based on the derivative or slope of the error. This term decreases the inputs whenever the error gets smaller, and increases the inputs when the error gets larger. This term can mitigate any overshoot effects caused by the P and I gains [27].

A PID controller is only applicable to Single-Input Single-Output (SISO) systems. For this reason it is not directly applicable to underwater vehicles with multiple thrusters, especially if the underwater vehicle has coupled dynamics [28]. To tackle this the PID controllers are usually applied to uncoupled systems, where they operate only on 1 DOF. In order to operate multiple DOF multiple PID controllers have to be designed and tuned for each DOF [29] [30]. Here they might also choose to control the AUV only in the heave, yaw and surge directions. This leads to satisfactory results when following a trajectory when objects are not closeby and very precise control is not needed [31].

Another approach when using PID controllers in systems that are coupled or overactuated is to make use of an in-between step [19]. Here, the PID controller computes a vector with the forces and moments in 6 DoF, similar to $\boldsymbol{\tau}$. This so-called wrench vector is then transformed into thruster inputs via an optimization step, also known as a thruster allocation algorithm [32]. This computes the inputs while trying to minimize two terms: the error to the wrench vector in order to follow the reference and the input to the thrusters in order to be as efficient as possible. It does so while adhering to the thruster constraints.

There can be situations where the AUV or the environment it is in is changing. Think for example of a sudden disturbance via a water current or waves, or the addition of a manipulator arm or other type of hull to the AUV. In these situations a classic PID controller might not lead to satisfactory results anymore [33] [34]. This is mainly because the PID controller is not tuned correctly anymore for the changed environments. In order to tackle this it is possible to make use of a self-tuning PID controller. These self-tuning controllers can make use of neural networks [34], fuzzy control techniques [35] or inverse optimal control technique [36] to continuously update the weights of their PID controllers while operating. This leads to a better performance under changing conditions compared to classic PID control. As deviations in water currents can be expected in the AUV inspection robot during quay wall inspection this might prove a valuable solution to its control problem.

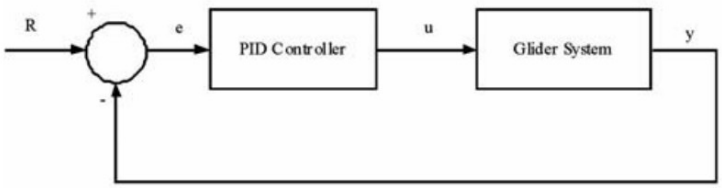


Figure 11: PID scheme (glider system = AUV) [37]

5.2 Model based Control

In contrast to the aforementioned classical controllers a model based controller uses a model of the system to compute the plant control inputs. A common approach to model the plant is via the use of a state space representation of the system. A linear state space representation of an AUV is explained in Section 4. In this Section two types of model based controllers are described, a Linear Quadratic Regulator controller (LQR) and a Model Based Predictive controller (MPC).

5.2.1 Linear Quadratic Regulator Controller

A Linear Quadratic Regulator controller uses a state space representation of the system to compute an optimal feedback gain K by minimizing a quadratic cost function J, see Figure 12 . This state space representation of the system is the one found in Section 4, equation 33. The cost function j is given by:

$$J = \int_0^\infty x(t)^\top Q x(t) + u(t)^\top R u(t) dt \tag{37}$$

In this cost function J both Q and R are positive semi-definite matrices, where Q is a matrix that penalizes the state deviation x and R is a matrix that penalizes the control input u. These matrices need to be determined iteratively, where a good balance needs to be found in allowance of a state deviation and control efforts [37]. In the case of an inspection robot close to quay walls it would be desirable to choose a high penalizing factor on state deviations in the direction of the quay wall, as to not hit it. The objective of the LQR controller is to derive the feedback gain K as $u = -Kx$ [38]. In order to account for errors in the steady state of the system it is necessary to make use of a pre-compensator. In Figure 12 this pre-compensator is denoted by $NBar$.

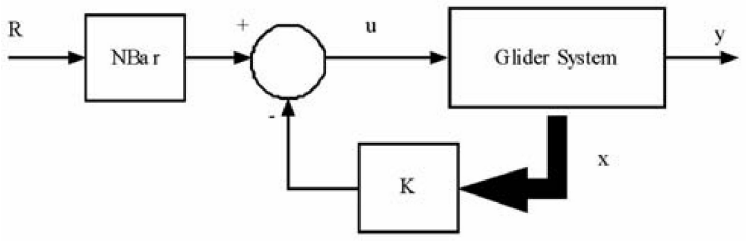


Figure 12: LQR scheme (glider system = AUV) [37]

LQR control has proven to have an improved performance over PID, PD and PI control in glider shaped AUV [37] [38]. LQR has also proven to have sufficient performance in the case of a spherical shape AUV, see [39]. Here, it is also suggested that an MPC controller might have an improved performance over LQR control. In order to improve LQR control for AUV a LQR PI vectorial (Linear Quadratic Regulator Proportional Integral) vectorial design of a controller has been attempted [40]. This added PI vectorial action reduces the steady-state error due to a disturbance to the pre-compensator due to a variation of the system. This makes the system more robust to real-life applications where the true model is not exactly known. This LQR PI vectorial controller shows a good performance in the path-tracking of an underwater vehicle in simulation. As the controller has not yet been tested in experiments it remains unknown if the controller is sufficient in real-life applications as well. Another possible improvement over conventional LQR control is the use of optimal LQR control with the use of a Multiobjectives Differential Evolution (MODE) approach [41]. Here, the MODE approach computes Pareto-based optimal (sub-optimal) parameters of the controller, taking multiple objectives into account. This is done to find a balance between conflicting objectives, such as time response and actuation energy. This approach proves to give a good performance in the presence of sensor noise and disturbances to the input of the system.

5.2.2 Model Predictive Control

A model predictive controller is a controller that uses a model of the system to make a finite time horizon prediction on the future states of the system and computes optimal control outputs by minimizing a cost function, see Figure 13. Over the last years it has proven to be a popular choice for controlling systems. This is mainly due to its ability to be able to handle constraints on the system, it is able to handle MIMO systems, it is able to take into account future reference information, and it generally has a good performance. A MPC controller works in three stages: prediction, optimization, and the receding horizon principle [42].

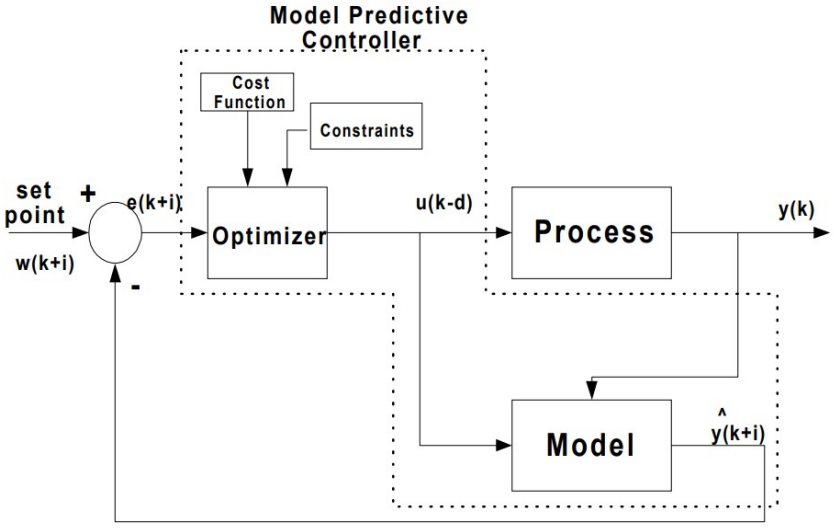


Figure 13: Model Predictive Control scheme [42]

5.2.2.1 Prediction

In the first stage of a MPC controller the controller computes the systems states along a finite future time horizon for possible control actions. This is done by using a discrete model of the system with a fixed sampling time interval. The trajectories of the system are computed over a finite prediction horizon, starting from an initial position, see Figure 14.

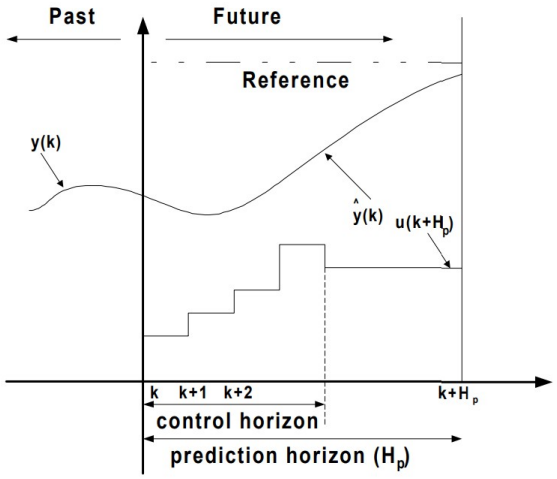


Figure 14: Model Predictive Control prediction scheme [42]

5.2.2.2 Optimization

In the optimization step of the MPC algorithm the optimal control input sequence is computed. This step consists of two parts: the cost function that has to be minimised and the constraints that have to be adhered to

In order to compute the optimal control action the MPC algorithm consists of an optimization step. In the optimization step the MPC algorithm aims to minimize a certain cost function. These cost functions can be defined and tuned differently for each system. Examples of cost functions used in underwater vehicles are given below [14] [21] [43].

$$J = \sum_{j=1}^{K} \Lambda_d - \Lambda_j(\mu_j)^2 + (\mu_j)^2 \tag{38}$$

Equation 38 is a cost function of the sum of squared distances between the desired state, Λ_d, and the predicted state, Λ_j. Here K is the horizon length, j is the current time step and $(\mu_j)^2$ is introduced to minimize energy usage.

$$J = \sum_{x \in \chi_k} x^\top \boldsymbol{Q} x + \sum_{u \in U_k} u^\top \boldsymbol{R} u \tag{39}$$

Equation 39 is a cost function of the combined sum of the deviation of the state, given by x, and the control effort, given by u. Matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive semi-definite matrices that can be tuned to balance the focus of the optimizer between state deviations and control effort. This cost functions is a quadratic function which can be solved with quadratic programming optimization algorithms, which are relatively computationally inexpensive.

In order for the optimization algorithm to not compute control actions that are not physically possible on the system the optimization step has to adhere to certain constraints. These constraints can be set on both the state and the control inputs and can be both equality and inequality constraints, and are given by:

$$\begin{aligned} \underline{x} \leq x \leq \overline{x} \quad & \forall x \in X_k \\ \underline{u} \leq u \leq \overline{u} \quad & \forall u \in U_k \end{aligned} \tag{40}$$

The complete optimization problem of the MPC controller is given by equation 41 [43].

$$\begin{aligned} & arg \min_{U_k} \quad J(X_k, U_k) \\ s.t. \quad & \begin{cases} x[j+1|k] = Ax[j|k] + Bu[j|k] & \forall j = k...k+N \\ x[k|k] = x[k] \\ \underline{x} \leq x \leq \overline{x} & \forall x \in X_k \\ \underline{u} \leq u \leq \overline{u} & \forall u \in U_k \end{cases} \end{aligned} \tag{41}$$

5.2.2.3 Receding Horizon Principle

At each timestep the MPC controller computes the optimal control action sequence for that finite time horizon in the optimization step. Next, it applies the first action of the sequence to the system, after which it moves to the next time step and starts again. This principle is called the receding horizon principle, and can also be seen in Figure 14. With this principle the algorithm always applies the optimal input to the system at each time step.

5.2.2.4 MPC Performance

Early applications of MPC to AUV systems show already good performance, see [42] [44]. Here MPC is only used to control 1 DOF as computation power is limited, and MPC control tends to be computationally expensive. It does however forecast the potential of MPC control when computational power is less of an issue, see [12]. Here MPC outperforms conventional feedback controllers in tracking error with over 40%. However, it also shows the sensitivity to inaccurate model parameter approximations and the inaccuracies of the linearized state space model.

To tackle this problem several attempts have been made to use Nonlinear Model Predictive Control (NMPC) to control an AUV [45] [46]. At the cost of more computational effort these controllers prove to have a better

performance than conventional MPC and also PID and LQR control methods. Within the NMPC methods it is also possible to add a disturbance to the state space model of the system, and have the NMPC controller reject this disturbance.

An even more promising approach to the control of an AUV is by using a Lyapunov-Based Model Predictive Control (LMPC) [47] [48] [49]. In this LMPC method the closed-loop stability and recursive feasibility are proved. This is a very desirable trait for the underwater inspection robot, as an instability in the control system can cause the AUV to crash into the quay wall, creating considerable damage to the AUV. Furthermore the LMPC method proves to be robust, where model parameter errors of up to 30% still give good performance and water currents of $100N$ still give a good performance. These are again very desirable traits, as it is expected that it will be difficult to accurately derive the model parameter of the inspection robot, as well as the fact that water currents can suddenly be present at the quay wall location that can now be rejected. Another advantage of the LMPC method is that it can simultaneously address the thrust allocation subproblem, as the thrust distribution can be directly formulated into the optimization problem. LMPC can also circumvent the local linearization step that is need by standard MPC control design to follow a curve trajectory.

5.3 Discussion

Classic control techniques such as Bang-Bang control and PID control have proven to be applicable to AUV. They do not need a model in order to be implemented, which makes the design of the controller easier. Regarding performance of the controllers, especially PID shows sufficient performance for AUV. However, model based controllers like LQR and MPC have proven to have an even better performance. They do require an accurate model of the AUV, but as the AUV will be tested in simulation a model of the AUV is needed anyway. In the performance of the model based controllers MPC proves to have a better performance over LQR control. It can be concluded that a MPC controller is a suitable choice for a controller design for an AUV. If stability, robustness and even better performance is desired as well a LMPC method is recommended.

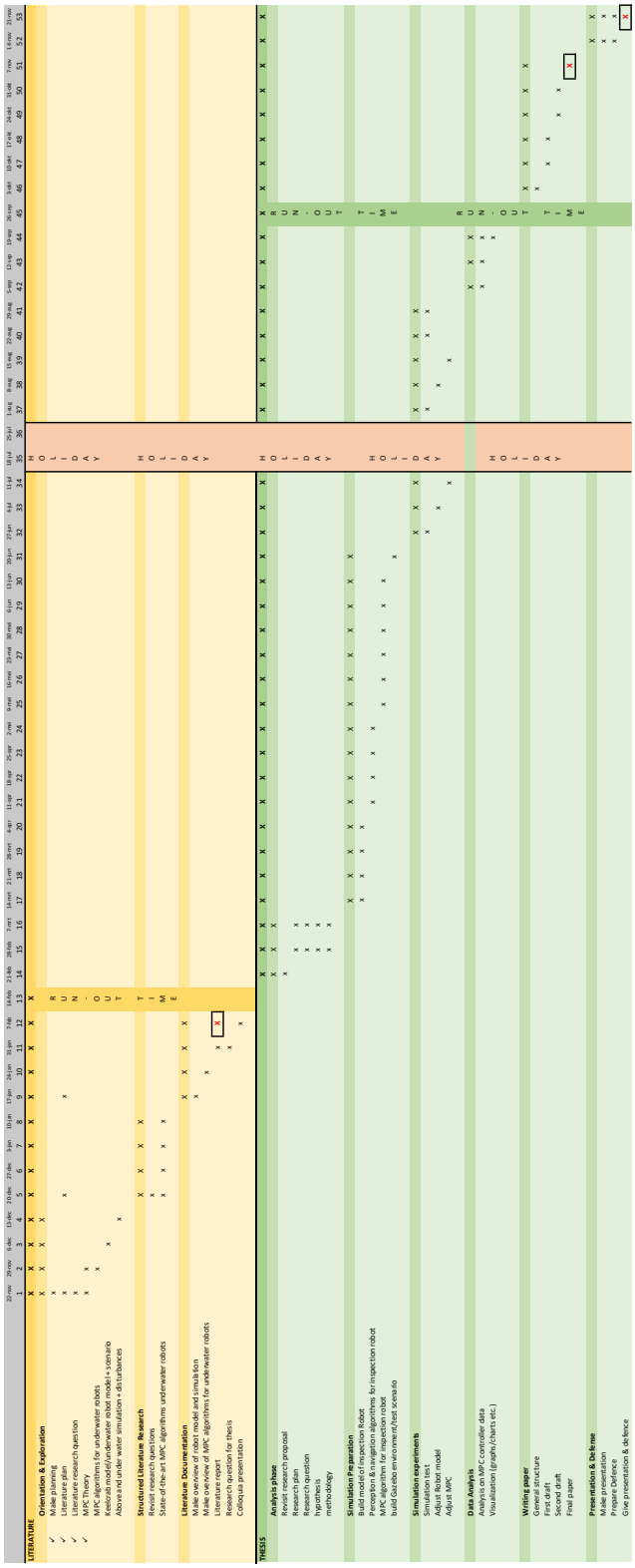
6 Conclusion and thesis approach

The goal of this literature review was to derive which control method was most suited for the control of a robot that performs the mission of a quay wall inspection. After considering the control methods Bang-Bang control, PID control, LQR control and MPC it can be concluded that MPC is the most suited for an AUV. This control method can best be designed and implemented using the Gazebo simulation software with the Plankton plugin, which can simulate the robot with Fossen's equations of motion for the robot. The MPC controller can be built in ROS2.

The thesis work should continue with the design of the robot and the way it will tackle the inspection mission. Several approaches regarding paths along the quay wall and robot thruster layouts are possible, and via simulation it can be determined which is the most efficient. When this design has been determined the next step is to determine the parameters of the model of the robot. This can be done numerically via software, or by doing experiments. Both should be attempted, where the experiments can be used to validate the numerical approach. The numerical approach will allow for faster prototyping as it will be easier to alter the design in later stages. The experiments will be conducted in a water tank. As there is no motion capture system present in the water tank an attempt will be made to localize the robot using camera's and fiducial markers on the robot. This is a cheap and relatively simple solution which hopefully proves to lead to accurate localization results of the robot. With the correct model parameters the robot can be accurately simulated in the Plankton plugin of the Gazebo simulator.

After the design, modelling and simulation of the robot it is time to design the MPC controller. This will be done in ROS2. As LQR can be considered as a more simple version of a MPC controller without constraints and with a infinite time horizon first an attempt will be made to design and implement an LQR controller. Afterwards a MPC controller will be built. If time allows for it an attempt will be done to implement the Lyapunov-based MPC controller. These controllers will all be built and tested in simulation. Depending on the speed of the project and whether or not a real prototype of the robot is available they will also be tested experimentally in the water tank. The results of the thesis will consist of a comparison between the given reference trajectory and the actual trajectories of the robot, which can be denoted as the performance of the controller.

7 Planning


LITERATURE
Orientation & Exploration
Make planning
Literature plan
Literature research question
MPC Theory
MPC algorithms for underwater robots
Koelzsch model/underwater robot model + scenario
Alcocer and Oliveir water simulation + disturbances
Structured Literature Research
Revisit research questions
State-of-the-art MPC algorithms underwater robots
Literature Documentation
Make overview of robot model and simulation
Make overview of MPC algorithms for underwater robots
Literature report
Research questions for thesis
Colloquia presentation
THESIS
Analysis phase
Research question proposal
Research plan
Research question
hypothesis
methodology
Simulation Preparation
Build model of Inspection Robot
Perception & navigation algorithms for inspection robot
Make overview of MPC algorithms for robot
build Gazebo environment/test scenario
Simulation experiments
Setup test
Adjust robot model
Adjust MPC
Data Analysis
Analysis of MPC controller data
Visualization (graphs/charts etc.)
Writing paper
Create structure
First draft
Second draft
Final paper
Presentation & Defence
Make presentation
Prepare Defence
Give presentation & defence

References

[1] N. Kishi, A. Asada, K. Abukawa, and K. Fujisawa, "Inspection methods for underwater structures of ports and harbors," in *2015 IEEE Underwater Technology (UT)*. IEEE, 2015, pp. 1–5.

[2] W. COMMUNICATION. Grimburgwal provides lessons for quay wall renovations amsterdam. [Online]. Available: https://www.tudelft.nl/en/2021/tu-delft/grimburgwal-provides-lessons-for-quay-wall-renovations-amsterdam

[3] C. Griwodz, S. Gasparini, L. Calvet, P. Gurdjos, F. Castan, B. Maujean, G. De Lillo, and Y. Lanthony, "Alicevision meshroom: An open-source 3d reconstruction pipeline," in *Proceedings of the 12th ACM Multimedia Systems Conference*, 2021, pp. 241–247.

[4] C. Beall, B. J. Lawrence, V. Ila, and F. Dellaert, "3d reconstruction of underwater structures," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4418–4423.

[5] P. Buschinelli, J. Salazar, D. Regner, D. Oliveira, M. Machado, G. Marcellino, D. Sales, J. Santos, C. Marinho, M. Stemmer *et al.*, "Targetless photogrammetry network simulation for inspection planning in oil and gas industry," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, pp. 285–291, 2020.

[6] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[7] I. Ugorji. Sheet pile walls and their uses. [Online]. Available: https://structville.com/2020/10/sheet-pile-walls-and-their-uses.html

[8] D.-H. Ji, H.-S. Choi, S.-K. Jeong, J.-Y. Oh, K. Seo-Kang, and S.-S. You, "A study on heading and attitude estimation of underwater track vehicle," *Advances in Technology Innovation*, vol. 4, no. 2, p. 84, 2019.

[9] N.-D. Nguyen and S.-K. Kim, "Navigation and control of underwater tracked vehicle using ultrashort baseline and ring laser gyro sensors," *Sens. Mater*, vol. 31, pp. 1575–1587, 2019.

[10] S. Hong, J.-S. Choi, H.-W. Kim, M.-C. Won, S.-C. Shin, J.-S. Rhee, and H.-u. Park, "A path tracking control algorithm for underwater mining vehicles," *Journal of mechanical science and technology*, vol. 23, no. 8, pp. 2030–2037, 2009.

[11] T.-K. Yeu, S.-M. Yoon, S.-J. Park, H.-W. Kim, C.-H. Lee, J.-S. Choi, K.-Y. Sung *et al.*, "Study on path tracking approach for underwater mining robot," in *2012 Oceans-Yeosu*. IEEE, 2012, pp. 1–5.

[12] A. Molero, R. Dunia, J. Cappelletto, and G. Fernandez, "Model predictive control of remotely operated underwater vehicles," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 2058–2063.

[13] V. Berg, "Development and commissioning of a dp system for rov sf 30k," Master's thesis, Institutt for marin teknikk, 2012.

[14] K. L. Walker, R. Gabl, S. Aracri, Y. Cao, A. A. Stokes, A. Kiprakis, and F. Giorgio-Serchi, "Experimental validation of wave induced disturbances for predictive station keeping of a remotely operated vehicle," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5421–5428, 2021.

[15] A. S.R.L. Keelcrab for boatowners. [Online]. Available: https://www.keelcrab.com/en/boat/the-keelcrab

[16] R. González, F. Rodríguez, and J. L. Guzmán, "Autonomous tracked robots in planar off-road conditions," in *Modelling, Localization, and Motion Control. 2014, Modelling, Localization and Motion Control*. Springer, 2014.

[17] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[18] O. Faltinsen, *Sea loads on ships and offshore structures*. Cambridge university press, 1993, vol. 1.

[19] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation," in *OCEANS 2016 MTS/IEEE Monterey*. IEEE, sep 2016. [Online]. Available: https://doi.org/10.1109%2Foceans.2016.7761080

[20] M. Pecka, K. Zimmermann, and T. Svoboda, "Fast simulation of vehicles with non-deformable tracks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6414–6419.

[21] D. C. Fernández and G. A. Hollinger, "Model predictive control for underwater robots in ocean waves," *IEEE Robotics and Automation letters*, vol. 2, no. 1, pp. 88–95, 2016.

[22] M. Chyba, H. Sussmann, H. Maurer, and G. Vossen, "Underwater vehicles: The minimum time problem," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 2. IEEE, 2004, pp. 1370–1375.

[23] J.-Y. Park, B.-H. Jun, P.-M. Lee, J.-H. Oh, and Y.-K. Lim, "Underwater docking approach of an under-actuated auv in the presence of constant ocean current," *IFAC proceedings volumes*, vol. 43, no. 20, pp. 5–10, 2010.

[24] S. Ziaeefard, B. R. Page, A. J. Pinar, and N. Mahmoudian, "A novel roll mechanism to increase maneuverability of autonomous underwater vehicles in shallow water," in *OCEANS 2016 MTS/IEEE Monterey*. IEEE, 2016, pp. 1–5.

[25] S. Smith, G. Rae, D. Anderson, and A. Shein, "Fuzzy logic control of an autonomous underwater vehicle," *Control Engineering Practice*, vol. 2, no. 2, pp. 321–331, 1994.

[26] M. Chyba, S. Grammatico, V. T. Huynh, J. Marriott, B. Piccoli, and R. N. Smith, "Reducing actuator switchings for motion control of autonomous underwater vehicles," in *2013 American Control Conference*. IEEE, 2013, pp. 1406–1411.

[27] F. Kong, Y. Guo, and W. Lyu, "Dynamics modeling and motion control of an new unmanned underwater vehicle," *IEEE Access*, vol. 8, pp. 30 119–30 126, 2020.

[28] Ö. Yildiz, R. B. Gökalp, and A. E. Yilmaz, "A review on motion control of the underwater vehicles," in *2009 International Conference on Electrical and Electronics Engineering-ELECO 2009*. IEEE, 2009, pp. II–337.

[29] I. Carlucho, B. Menna, M. De Paula, and G. G. Acosta, "Comparison of a pid controller versus a lqg controller for an autonomous underwater vehicle," in *2016 3rd IEEE/OES South American International Symposium on Oceanic Engineering (SAISOE)*. IEEE, 2016, pp. 1–6.

[30] V. Upadhyay, S. Gupta, A. Dubey, M. Rao, P. Siddhartha, V. Gupta, S. George, R. Bobba, R. Sirikonda, A. Maloo *et al.*, "Design and motion control of autonomous underwater vehicle, amogh," in *2015 IEEE Underwater Technology (UT)*. IEEE, 2015, pp. 1–9.

[31] X. Xiang, D. Chen, C. Yu, and L. Ma, "Coordinated 3d path following for autonomous underwater vehicles via classic pid controller," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 327–332, 2013.

[32] T. A. Johansen, T. I. Fossen, and P. Tøndel, "Efficient optimal constrained control allocation via multiparametric programming," *Journal of guidance, control, and dynamics*, vol. 28, no. 3, pp. 506–515, 2005.

[33] K. L. Walker, A. A. Stokes, A. Kiprakis, and F. Giorgio-Serchi, "Investigating pid control for station keeping rovs," in *Proceedings of the UKRAS20 Conference:"Robots into the real world", Lincoln, UK*, 2020, pp. 51–53.

[34] R. Hernández-Alvarado, L. G. García-Valdovinos, T. Salgado-Jiménez, A. Gómez-Espinosa, and F. Fonseca-Navarro, "Neural network-based self-tuning pid control for underwater vehicles," *Sensors*, vol. 16, no. 9, p. 1429, 2016.

[35] X. Xiang, C. Yu, L. Lapierre, J. Zhang, and Q. Zhang, "Survey on fuzzy-logic-based guidance and control of marine surface vehicles and underwater vehicles," *International Journal of Fuzzy Systems*, vol. 20, no. 2, pp. 572–586, 2018.

[36] R. Rout and B. Subudhi, "Inverse optimal self-tuning pid control design for an autonomous underwater vehicle," *International Journal of Systems Science*, vol. 48, no. 2, pp. 367–375, 2017.

[37] M. M. Noh, M. R. Arshad, and R. M. Mokhtar, "Depth and pitch control of usm underwater glider: Performance comparison pid vs. lqr," 2011.

[38] D. A. Lakhwani and D. M. Adhyaru, "Performance comparison of pd, pi and lqr controller of autonomous under water vehicle," in *2013 Nirma University International Conference on Engineering (NUiCONE)*. IEEE, 2013, pp. 1–6.

[39] R. A. S. Fernandez, D. Grande, A. Martins, L. Bascetta, S. Dominguez, and C. Rossi, "Modeling and control of underwater mine explorer robot ux-1," *IEEE Access*, vol. 7, pp. 39 432–39 447, 2019.

[40] S. Rodríguez, M. Peña, and R. Ramírez, "Dynamic control design lqr pi vectorial of remotely operated underwater vehicle," in *2014 III International Congress of Engineering Mechatronics and Automation (CIIMA)*. IEEE, 2014, pp. 1–4.

[41] I. B. Tijani and A. Budiyono, "Control of an unmmaned underwater vehicles using an optimized lqr method," *Marine and Underwater Science and Technology, ISIUS*, vol. 1, no. 1, pp. 41–48, 2016.

[42] W. Naeem, "Model predictive control of an autonomous underwater vehicle," in *Proceedings of UKACC 2002 Postgraduate Symposium, Sheffield, UK, September*. Citeseer, 2002, pp. 19–23.

[43] C. E. S. Koch, "Model predictive control for six degrees-of-freedom station-keeping of an underwater vehicle-manipulator system," 2017.

[44] A. Budiyono, "Model predictive control for autonomous underwater vehicle," 2011.

[45] S. Heshmati-Alamdari, G. C. Karras, P. Marantos, and K. J. Kyriakopoulos, "A robust model predictive control approach for autonomous underwater vehicles operating in a constrained workspace," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6183–6188.

[46] Y. Cao, B. Li, Q. Li, A. A. Stokes, D. M. Ingram, and A. Kiprakis, "A nonlinear model predictive controller for remotely operated underwater vehicles with disturbance rejection," *IEEE Access*, vol. 8, pp. 158 622–158 634, 2020.

[47] C. Shen, Y. Shi, and B. Buckham, "Trajectory tracking control of an autonomous underwater vehicle using lyapunov-based model predictive control," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5796–5805, 2017.

[48] ——, "Lyapunov-based model predictive control for dynamic positioning of autonomous underwater vehicles," in *2017 IEEE International Conference on Unmanned Systems (ICUS)*. IEEE, 2017, pp. 588–593.

[49] P. Gong, Z. Yan, W. Zhang, and J. Tang, "Lyapunov-based model predictive control trajectory tracking for an autonomous underwater vehicle with external disturbances," *Ocean Engineering*, vol. 232, p. 109010, 2021.