3D AUV Collision Avoidance

Øystein Engelhardtsen

Master of Science in Engineering Cybernetics
Submission date: June 2007
Supervisor: Kristin Ytterstad Pettersen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

Collision avoidance is a crucial part of an AUV's ability to perform planned missions in unknown environments. The assignment is to create a system for 3D collision avoidance maneuvering for the HUGIN AUV, able to handle vertical obstacles from ocean floor to surface (e.g. oil rig foundations or piers) and rough sea floor topography in combination with surface ice.

1. Do a literature survey and give and overview of state-of-the-art technology on AUV collision avoidance in 2 and 3 dimensions.

2.Develop an AUV simulator based on a complete hydrodynamic modell of the HUGIN AUV, including a guidance and navigation system, sensor measurements, collision avoidance system and a terrain consisting of vertical obstacles, rough sea floor topography and surface ice.

3.Develop a system for collision avoidance and justify chosen solutions. Assume the following measurements are available to the system:

o Vertical sonar, measuring altitude from the AUV to the ocean floor.

o Forward tilted sonar measuring upcomming altitude to the ocean floor

o Right- and left side-scan-sonars, oriented perpendicularly to the AUV heading, measuring distances to passing objects.

o 2D Forward Looking Sonar (line array) with 45 sonar beams covering a sector of 45O, each covering 1 O x15 O with a range of 50 m providing distances for each of the 45 sonar beams.

o Other availiable sensors on the HUGIN AUV that prove to be useful for collision avoidance purposes may also be included.

4.Discuss the derived system with respect to stability and its ability to avoid collisions.

5.Construct a mission plan where the AUV meets all the obstacles in item 2. Demonstrate a collision avoidance system which brings the AUV from start point to endpoint. Vary the distribution of obstacles to check the robustness of the system.

Assignment given: 09. January 2007
Supervisor: Kristin Ytterstad Pettersen, ITK

Preface

This master thesis is written as a compulsory part of the Master of Technology program on NTNU, department of engineering cybernetics. The project work on AUV collision avoidance has certainly been hard and consumed a great deal of time. Even so, the work has been truly interesting and rewarding. The available literature on this subject seems endless, still millions of dollars are spent on R&D around the world, and the AUV collision avoidance community is constantly increasing.

I would like to thank all my guidance supervisors for the help during all the phases of the project. A special thanks to Øivind Midtgaard at FFI and Bjørn Jalving at Kongsberg Maritime, for always politely replying to my endless series of questions regarding the HUGIN AUV.

Øystein Engelhardtsen
Trondheim, June 2007

Abstract

An underlying requirement for any Autonomous Underwater Vehicle (AUV) is to navigate through unknown or partly unknown environments while performing certain user specified tasks. The loss of an AUV due to collision is unjustifiable both in terms of cost and replacement time. To prevent such an unfortunate event, one requires a robust and effective Collision Avoidance System (CAS). This paper discusses the collision avoidance problem for the HUGIN AUVs. In the first part, a complete simulator for the HUGIN AUV is implemented in matlab and simulink. This includes a 6 degrees-of-freedom nonlinear AUV model, simulated environment including bottom profile and surface ice, navigation- and guidance functionality and sensor simulators. In the second part a number of well known strategies for the collision avoidance problem is presented with a short analysis of their properties. On the basis of the implemented simulator, a proposed CAS is developed and it's performance is analyzed. This system is based on simple principles and known collision avoidance strategies, in order to provide effective and robust performance. The proposed system provides feasible solutions during all simulations and the collision avoidance maneuvers are performed in accordance with the specified user demands. The developed simulator and collision avoidance system is expected to provide a suitable framework for further development and possibly a physical implementation on the HUGIN AUVs.

Contents

List of Tables

Abbreviations

AINS	Aided Inertial Navigation System
AUV	Autonomous Underwater Vehicle
CAS	Collision Avoidance System
CTD	Conductivity-Temperature-Density
DVL	Doppler Velocity Log
DTM	Digital Terrain Model
FLS	Forward Looking Sonar
HISAS	HIgh resolution Interferometric Synthetic Aperture Sonar
IMU	Inertial Measurement Unit
MBE	Multi-Beam Echo Sounder
MCM	Mine Counter Measures
MRS	Mine Reconnaissance System
REA	Rapid Environmental Assessment
SAS	Synthetic Aperture Sonar
SSS	Side-Scan Sonar

Chapter 1

Introduction

1.1 Motivation

In the past few decades, the number of Autonomous Underwater Vehicles (AUVs) has increased exponentially. This is largely because of their excellent ability to explore and assess the underwater environment. And as the technology development keeps progressing at a steady pace, the cost for AUVs are decreasing and becoming available to an equally increasing crowd. Whether it is mapping of the ocean floor for oil installations, assessing a naval mine threat or collection of oceanographic data, these vehicles provide users a great resource for better understanding of the ocean in general. (Tan et al. 2004*a*)

The most common way of planning an AUV mission today is to program a path based on previous knowledge of the environment, existing of a set of Way Points (WPs). A guidance and navigation functionality will then perform the task of following this predefined WP path. A limitation of most AUVs are their inability to adjust to sudden or previously unknown obstacles along this path, which can pose a threat in the form of a potential collision. Potential collision threats are foremost the ocean floor itself, especially in rugged terrain, but may also include a dredged harbor lane, an obstacle proud of the ocean floor, surface ice and ships maneuvering at the ocean surface. Although most AUVs are built to cope with a certain degree of impact, a collision will often lead to a mission-abort and may in some cases even cause damage to the vehicle or to the collision object. In any case, the inability to solve a potential collision threat and proceed on the planned mission will often lead to economical consequences in some degree. (Horner et al. 2005)

Recent advances in sonar technology has enabled the development of relatively low cost, low power, Forward Looking Sonars (FLS). These sensors have the ability to provide AUVs information on upcoming terrain and may be used for obstacle detection. FLSs come in many different formats and often have multiple sonar beams. By using this information in an appropriate way, an AUV could not only be able to detect an obstacle in the path, but also to calculate a way of avoiding it.(Horner et al. 2005) The AUV considered in this paper is equipped with a 2 dimensional FLS with 45 beams covering a sector of 45° with each of the beams covering a sector of 1° x 15°.

1.2 Project scope and emphasis

In this paper, collision avoidance is defined as the ability to avoid previously unknown obstacles on the predefined path while still attempting to accomplish the mission objective. As given in the assignment text, the operation environment is considered unknown. As a consequence of this, all obstacles on the path are initially unknown and collision avoidance maneuvers may only be activated once a potential collision is detected based on sensor information.

The complete set of requirements in the assignment text and the complexity of the given task, involves a workload that greatly extends the available time for this master thesis, if every sub-problem is to be fully analyzed and the developed system is to handle every thinkable situation. For this reason, the emphasis in this paper is put on developing a robust concept of collision avoidance and to demonstrate this concept in simulations.

The initial design presented in this paper assumes the following restrictions on the operation environment:

- There are no dynamical obstacles present
- All measurements are considered free of noise and errors
- There is no current

These restrictions allows for a development focusing on the core functionality of the system without delving to much with what is considered to be implementation issues. It is assumed that all of these restrictions may be partly- or fully removed later, by introducing added functionality based on the same principle design. Also, introducing these restrictions is not in conflict with the assignment text.

1.3 Paper layout

As this project covers a number of different and partly independent subjects, it does not follow the normal standard of technical reports. Instead, each subject is treated separately followed by simulations and discussions regarding that particular section. This layout was chosen to improve the readability of the report.

In the two first chapters the reader is introduces to the collision avoidance problem and the HUGIN AUVs. Part I covers the development of the AUV simulator, representing the HUGIN AUV as it is today. Part II deals with the collision avoidance problem and a concept for a collision avoidance is developed, implemented and simulated using the simulator from part I. Finally the proposed system is discussed and suggestions for future work is presented.

Chapter 2

The HUGIN AUVs

2.1 Background

The goal application for the system developed in this paper is the HUGIN AUV (*www.ffi.no/hugin* n.d.) , developed in collaboration between the Norwegian Defense Research Establishment and Kongsberg Maritime. By request, the collision avoidance system will be designed to be compatible with the existing functionality of the HUGIN AUV. Therefore, this section gives a short introduction to HUGIN, it's main operational tasks and functionality.

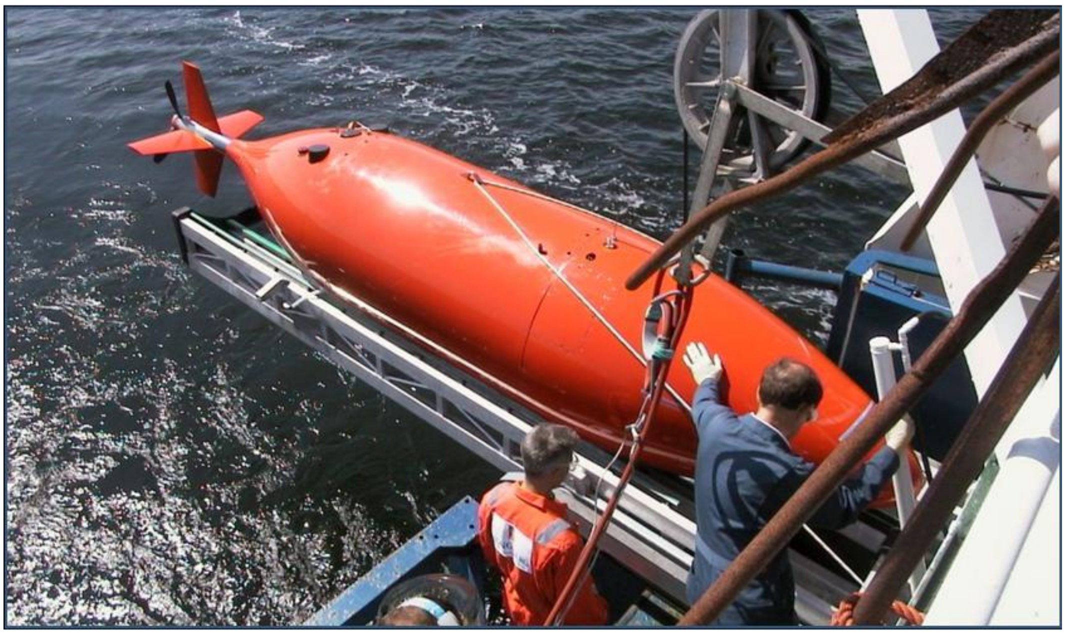


Figure 2.1: The HUGIN AUV

The development started in the 1990s and HUGIN vehicles are today used on a routine basis for mapping and imaging services within the offshore oil and gas industry , naval defense and ocean and fishery sciences. HUGIN has achieved great success in the offshore industry, and survey companies in Norway, USA and Holland currently own and operate HUGIN 3000 systems on a regular basis. A total of seven civilian HUGIN vehicles have been delivered, with an accumulated billed survey distance fast approaching 100,000 km. In addition, a specialized version (HUGIN MRS) has been developed for mine-counter measures (MCM) and rapid environmental assessment (REA) in naval operations. The MCM vessel KMN Karmøy has employed HUGIN vehicles during numerous national and international exercises since 2001. In 2004, a military pilot version (HUGIN 1000) was delivered to the Royal Norwegian Navy for permanent installation on board. (Midtgård et al. n.d.)

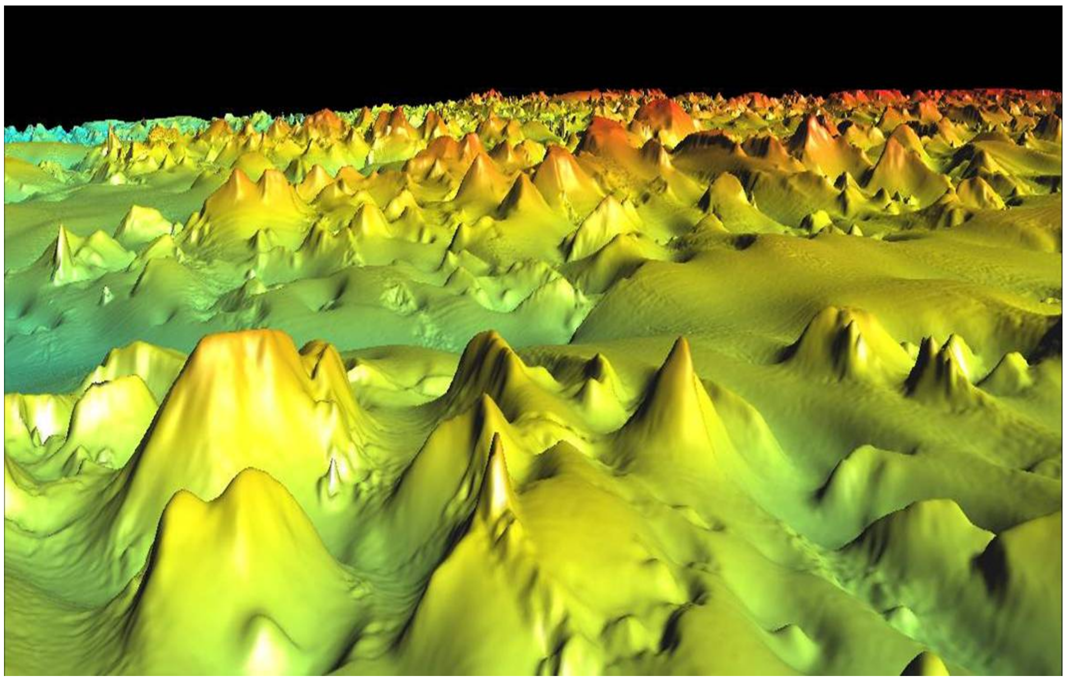


Figure 2.2: DTM from the Ormen Lange field, created by HUGIN. (Kongsberg-Maritime n.d.)

2.2 HUGIN 1000

The HUGIN 1000 represents the third generation of the HUGIN vehicles and builds upon the track-record accumulated through the commercial survey operations with the HUGIN I/II and HUGIN 3000 systems. The system is built around the same principles as HUGIN 3000, but is only half the volume, still maintaining the capability of carrying a sophisticated suite of sensor systems. HUGIN 1000 is primarily designed to meet the operational requirements from Military applications and Research and environmental monitoring applications. HUGIN 1000 can operate in two operational modes; acoustically supervised from an accompanying surface ship or fully autonomous. Autonomous mode is required for e.g.

covert military operations, mission under ice or to optimize efficiency by reliving the mother ship to perform other duties. A collision avoidance system is a crucial component for fully autonomous operation.

Figure 2.3: HUGIN 1000 (Kongsberg-Maritime n.d.)

HUGIN 1000 is currently equipped with an Aided Inertial Navigation System (AINS) including; Inertial Measurement Unit (IMU), Doppler Velocity Log (DVL), Pressure Sensor and Acoustic Navigation transponder. A path programmed in to HUGIN consist of a set of horizontal WP's, and given reference altitudes above the ocean floor on each leg. Additional sensors are also available for improved accuracy. The AINS keeps track of the current position in 3 dimension using a Kalman filter to combine inputs from the available sensors, while following the predefined path in the horizontal plane. Typical position accuracy range from 0.2% for to 0.01% of travelled distance, depending on sensor configuration and trajectory pattern. (Kongsberg-Maritime n.d.) The bottom following operation maximizes the use of the surveying sensors by keeping a constant altitude above the bottom. This capability is obtained through an altitude controller using feedback and feedforward from a separate altitude and forward-looking echo sounder system. The altitude controller is able to maintain a given altitude to an acceptable degree as long as the path does not meet any obstacles where a vertical avoidance is infeasible or the ocean floor is too rugged. Currently, no additional collision avoidance system is implemented, so in these cases human intervention is neccessary to avoid mission abortion and/or collision. (Midtgård et al. n.d.)

The HUGIN concept allows integration of alternative sensors for geophysical research and inspection purposes. Such sensors are called payload sensors and is handled by a separate payload processor on the vessel. In most cases, the basic survey package selected by the user will consist of all, or combinations, of the following sensors:

- synthetic aperture sonar (SAS) or side-scan sonar (SSS)
- multi beam echo sounder (MBE)
- sub-bottom profiler
- Conductivity-Temperature-Density (CTD) sensor
- volume search sonar

All of the payload sensors are mainly used to collect data about the passing environment and are not used directly for navigational purposes. However, some sensors (eg. MBE and later;ISAS) are able to build up a DTM of the covered area which could be useful when the AUV is passing through previously covered paths or nearby areas. This DTM may potentially be used to improve navigation accuracy by bathymetry recognition. Such information may also be used to re-plan the mission underway, producing a new collision free trajectory based on the known obstacles before they are encountered. This is particularly relevant when HUGIN is following 'lawn-mover' patterns, as shown in figure 2.4, which is a common way of operation. However, no such functionality is so far implemented on the HUGIN range of AUV's. (Kongsberg-Maritime n.d.)

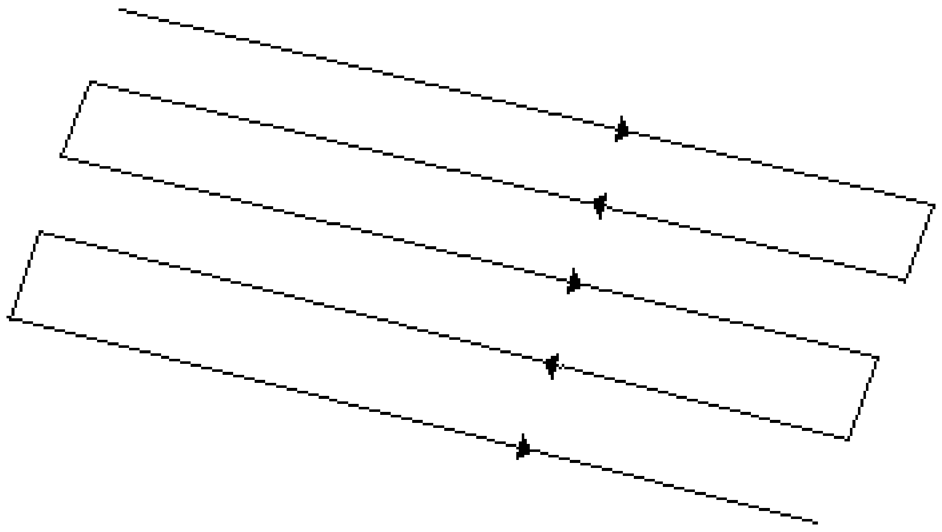


Figure 2.4: HUGIN operating in lawn-moving pattern

Operational depth:	0 to 1000 m
Length:	4 to 5 m
Maximum diameter:	0.75 m
Volume:	1.3 m^3
Weight in air:	approximately 650 kg
Weight in water:	Neutral
Nominal speed:	4 knots
Min / max speed:	2 to 5 knots
Total energy content of battery:	Modular, 3 to 15kWh
Vehicle endurance:	Up to 24 hours

Table 2.1: HUGIN 1000 - Basic vehicle specifications (Kongsberg-Maritime n.d.)

2.3 Navigation System

The navigation system of the HUGIN AUV is based on a decoupled structure where heading, pitch and speed are treated separately, much like the structure discussed in chapter 13 in (Fossen 2002). This decomposition has shown to describe the motion of slender formed vehicles like the HUGIN AUV quite accurate. This structure also allows for easy integration of a Collision Avoidance System (CAS) into the existing system architecture. Details about the HUGIN autopilot is kept confidential but guidance, navigation and control for AUVs has been addressed by a large number of authors. Some useful references are (Fossen 2002), (Fryxell et al. 1996) and (Leonard 1997)

For the decoupling to be valid, the roll angle should be kept close to zero at all times. Speed control is assumed a trivial task which may be performed by simple PID control or similar algorithms. The heading- and pitch control is somewhat more complicated and is covered by the two following sub-sections.

2.3.1 Heading control

The planned horizontal path, consisting of a set of WPs is programmed into the onboard computer. A guidance algorithm within the navigation system calculates a desired heading based on current position and the WP-path. This desired heading is fed into the heading autopilot which in turn performs the necessary rudder actuation. The guidance algorithm is assumed to be able to handle disturbances like currents and still be able to follow the planned path within an acceptable range. This sort of functionality is often called Trajectory Tracking and examples of such are PID cross-tracking, Line of Sight cross tracking and Linear Quadratic Optimal Cross-tracking. (Fossen 2002) An overview of the heading control system is given in figure 2.5.

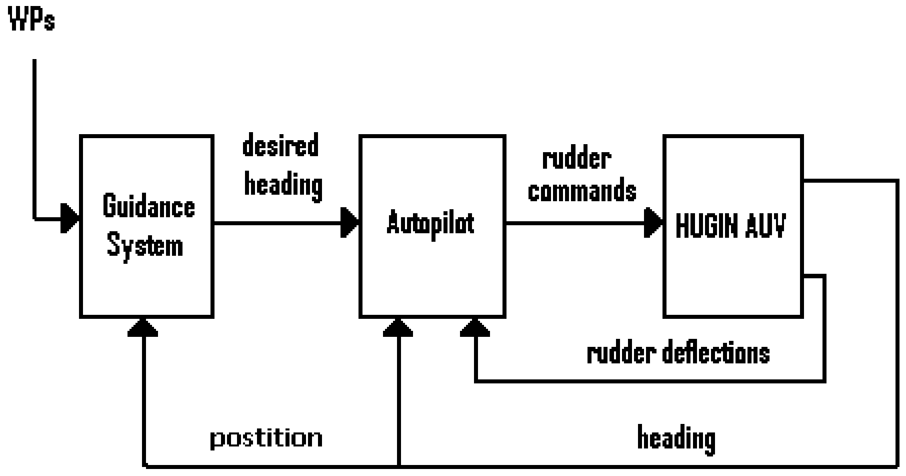


Figure 2.5: Functional architecture of the heading control system

2.3.2 Altitude control

As discussed earlier, the planned path also consist of a desired altitude above the ocean floor between two consecutive WPs. The pitch controller performs this functionality by feedback from a vertical echo sounder (vertical altimeter) and feedforward from a forward looking echo sounder (forward altimeter). The altitude control architecture consists of a pitch controller and an altitude controller. The altitude controller uses feedback from the vertical altimeter and feedforward from the forward altimeter to calculate the necessary pitch The echo sounder configuration is shown in figure 2.6 and the feedback/feedforward control principle is shown in figure 2.7.

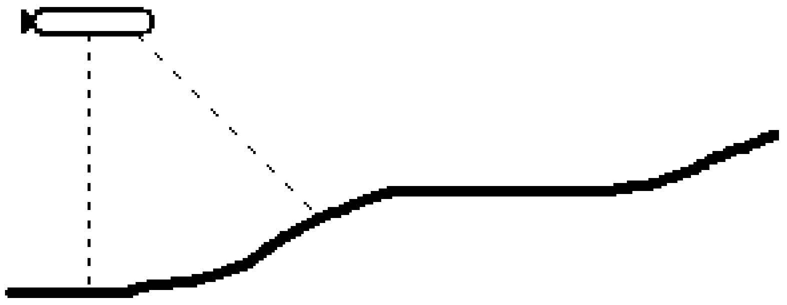


Figure 2.6: Altitude control echo sounder configuration

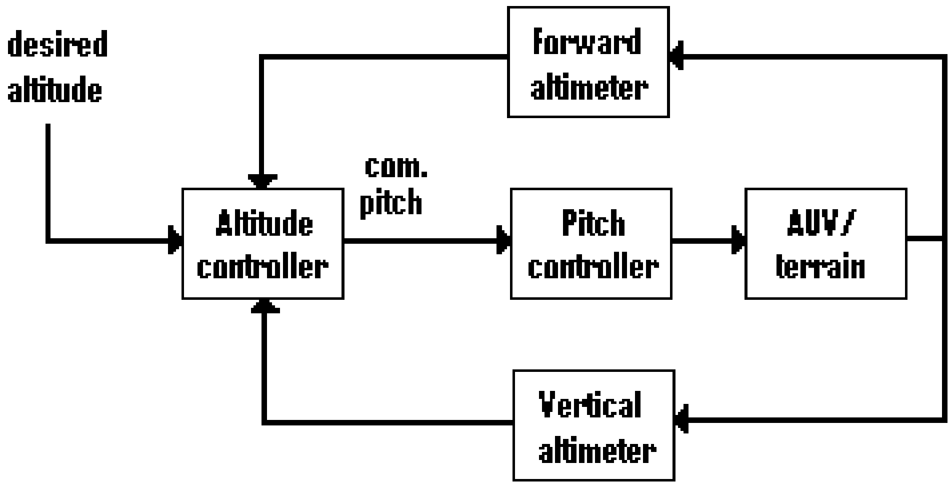


Figure 2.7: Altitude controller architecture

Part I

AUV Modelling and Control - Developing the AUV simulator

Simulator Overview

For the purpose of design and analyse of the proposed collision avoidance system (CAS), a suitable test-bed is desirable. For this reason, a complete simulation environment has been implemented in Simulink. A great deal of time and effort has been put into making this simulator as accurately and as close to reality as possible, to ensure maximum validity to the simulation results. The development of this simulator has also given the author a better grasp the problem in general and a better understanding of the physics and forces involved. The simulator includes a 6 Degrees of freedom nonlinear hydrodynamical model of the HUGIN 1000 AUV and simulators for each of the sonars using given specifications. Next an autopilot and a LOS Way point guidance system has been developed for the AUV model, representing the existing navigation system of the real vessel. Finally the bottom-following capability of the HUGIN AUVs needed to be replicated, so an altitude controller using the ranges from the sonar simulators was developed and implemented. Figure 2.8 shows an overview of the resulting simulator which is described in detail in this part.

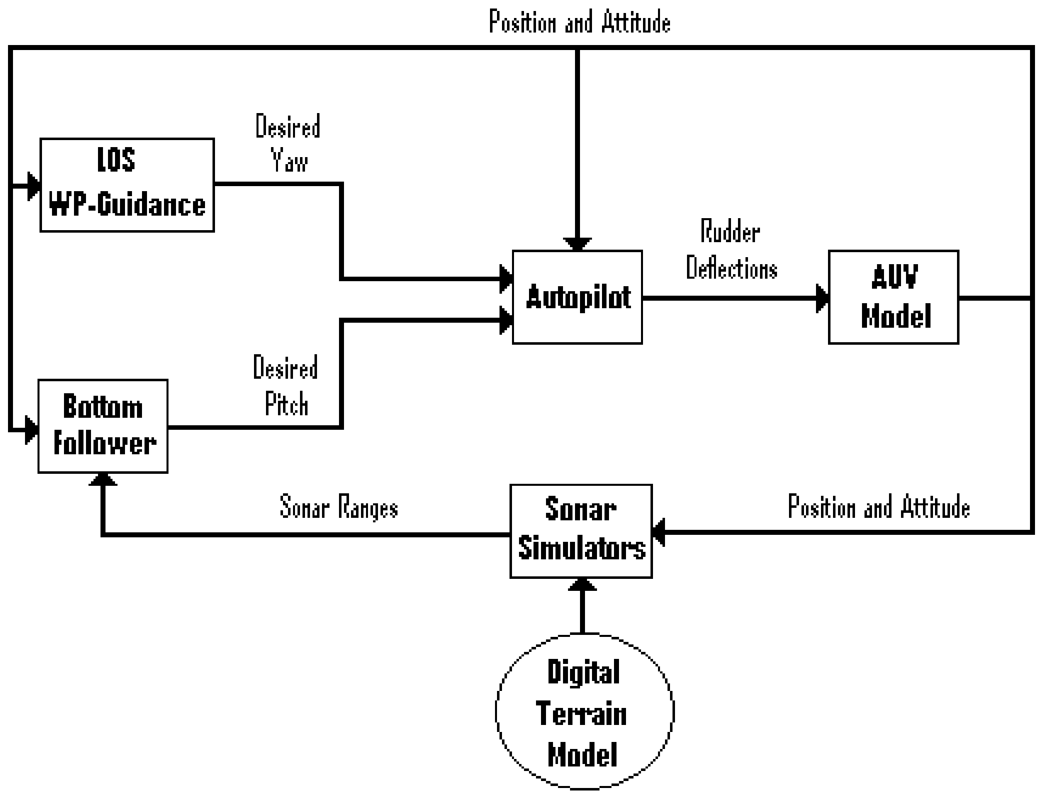


Figure 2.8: The AUV simulator structure

Chapter 3

Hydrodynamic modelling

This model is based on structure and parameters obtained from the development team behind the HUGIN AUVs (Kongsberg Maritime and FFI) and represents the most accurate (known) model of the HUGIN 1000 AUV to date. This is to ensure that simulations performed with the complete system is as close to reality as possible and reduce the need for adjustments in a real implementation. Still the structure of this model is of standard form for marine vessels and could represent any AUV, simply by using different model parameters and adjusting to propeller and rudder configurations.

3.1 Reference Frames and Motion Variables

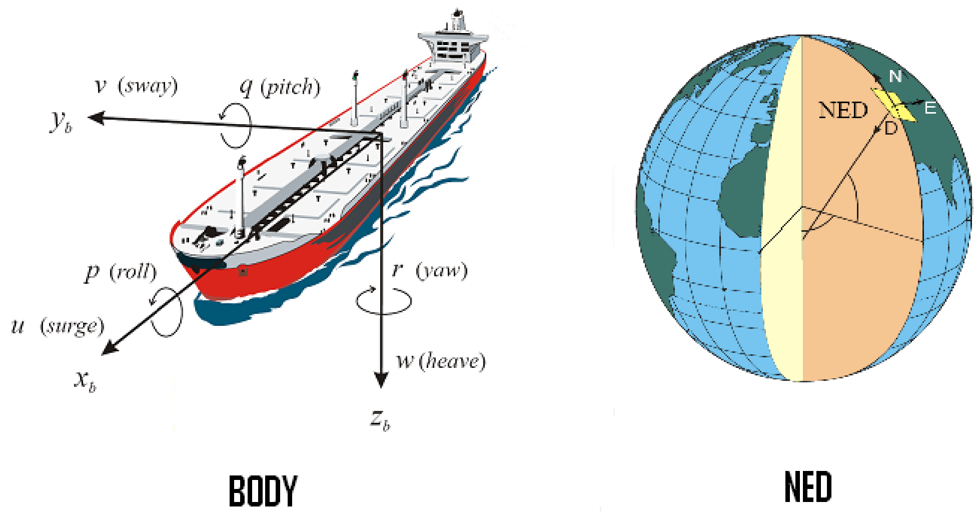


Figure 3.1: Reference frames (Fossen 2002)

For marine vessels operating in 6 degrees of freedom (DOF), 6 independent variables are necessary to determine position and orientation (attitude). The three first coordinates, and their time derivatives, correspond to position and translation. The last three coordinates, and their time derivatives, determines orientation and rotational motions (rates). For marine vessels, the 6 different motion components are defined as; surge, sway, heave, roll, pitch and yaw (see figure 3.1).

When analyzing the motion of a marine vessel within a limited geographic area, it is convenient to define two geographic reference frames; North-East-Down (NED) and Body frame (BODY). The NED frame is attached with the origin at a designated position on the earth surface. The X-component (N) corresponds with the local North direction, the Y-component corresponds with the local East direction and the Z-component points down along the normal of the earth surface (see figure 3.1). This frame is used for representing the position and orientation relative to the local tangent plane of the earth. I should be noted that this frame is not actually inertial since it moves and rotates with the earth, hence newtons laws does not apply perfectly in this frame. In addition, the earth curvature is not take into consideration. Still, for what is called flat-earth-navigation, the NED frame is normally considered inertial and geographically correct.

The BODY frame is a moving coordinate frame which is attached to the vessel hull. The position and orientation of the vessel are described relative to the reference frame NED while the linear and angular velocities are described in the BODY frame. The origin of the BODY frame is usually attached to the center of gravity or at the volumetric center of the vessel. For marine vessels the axes of the BODY frame are usually defined with the x-axis pointing from aft to fore, the-y axis pointing starboard and the z-axis pointing from top to bottom (see figure 3.1). For further details about reference frames for marine vessels, the reader is referred to Fossen (2002), chapter 2.

3.2 Developing the AUV model

3.2.1 Nonlinear hydrodynamic model

The dynamical equations of the system are based on the standard marine vessel equations of motion shown in equations 3.2.1 and 3.2.2, as described in Fossen (2002).

$$\dot{\eta} = J(\eta)\nu \tag{3.2.1}$$

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau + g_0 + w \tag{3.2.2}$$

$$\eta = \begin{bmatrix} x & (north) \\ y & (east) \\ z & (down) \\ \phi & (roll) \\ \theta & (pitch) \\ \psi & (yaw) \end{bmatrix}, \quad \nu = \begin{bmatrix} u & (surge) \\ v & (sway) \\ w & (heave) \\ p & (roll - rate) \\ q & (pitch - rate) \\ r & (yaw - rate) \end{bmatrix} \tag{3.2.3}$$

Equations 3.2.1 and 3.2.2 are actually 12 coupled differential equations, representing the dynamics of a vessel in 6 degrees of freedom (DOF). The matrices for equations 3.2.1 and 3.2.2 are defined in equations 3.2.4 to 3.2.12 where the notation of SNAME (1950) is used.

Mass Matrix:

$$
\begin{aligned}
M = M_{RB} + M_A = &\begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\ mz_g & 0-mx_g & -I_{yx} & I_y & -I_{yz} & \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z \end{bmatrix} \\
&+ \begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix}
\end{aligned} \tag{3.2.4}
$$

Coriolis and Centripetal Matrix:

$$C(\nu) = C_{RB}(\nu) + C_A(\nu) \tag{3.2.5}$$

where

$$
\begin{aligned}
C_{RB}(\nu) = &\left[\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -m(y_g + z_g r) & m(y_g p + w) & m(z_g p + v) \\ m(x_g q - w) & -m(z_g r + x_g p) & m(z_g p - v) \\ m(x_g r + v) & m(y_g r - u) & -m(x_g p + y_g q) \end{matrix}\right. \\
&\left.\begin{matrix} m(y_g q + z_g r) & -m(x_g q - w) & -m(x_g r + v) \\ -m(y_g p + w) & m(z_g r + x_g p) & -m(y_g r - u) \\ -m(z_g p - v) & -m(z_g q + u) & m(x_g p + y_g q) \\ 0 & -I_{yz} q - I_{xz} p + I_z r & I_{yz} r + I_{xy} p - I_y q \\ I_{yz} q + I_{xz} p + I_z r & 0 & -I_{xz} r - I_{xy} q + I_x p \\ -I_{yz} r - I_{xy} p + I_y q & I_{xz} r + I_{xy} q - I_x p & 0 \end{matrix}\right]
\end{aligned} \tag{3.2.6}
$$

and

$$
C_A(\nu) = \begin{bmatrix} 0 & 0 & 0 & 0 & -a_3 & a_2 \\ 0 & 0 & 0 & a_3 & 0 & -a_1 \\ 0 & 0 & 0 & -a_2 & a_1 & 0 \\ 0 & -a_3 & a_2 & 0 & -b_3 & b_2 \\ a_3 & 0 & -a_1 & b_3 & 0 & -b_1 \\ -a_2 & a_1 & 0 & -b_2 & b_1 & 0 \end{bmatrix} \tag{3.2.7}
$$

where

$$
\begin{aligned}
a_1 &= X_{\dot{u}} u + X_{\dot{v}} v + X_{\dot{w}} w + X_{\dot{p}} p + X_{\dot{q}} q + X_{\dot{r}} r \\
a_2 &= Y_{\dot{u}} u + Y_{\dot{v}} v + Y_{\dot{w}} w + Y_{\dot{p}} p + Y_{\dot{q}} q + Y_{\dot{r}} r \\
a_3 &= Z_{\dot{u}} u + Z_{\dot{v}} v + Z_{\dot{w}} w + Z_{\dot{p}} p + Z_{\dot{q}} q + Z_{\dot{r}} r \\
b_1 &= K_{\dot{u}} u + K_{\dot{v}} v + K_{\dot{w}} w + K_{\dot{p}} p + K_{\dot{q}} q + K_{\dot{r}} r \\
b_2 &= M_{\dot{u}} u + M_{\dot{v}} v + M_{\dot{w}} w + M_{\dot{p}} p + M_{\dot{q}} q + M_{\dot{r}} r \\
b_3 &= N_{\dot{u}} u + N_{\dot{v}} v + N_{\dot{w}} w + N_{\dot{p}} p + N_{\dot{q}} q + N_{\dot{r}} r
\end{aligned} \tag{3.2.8}
$$

Damping Matrix:

$$D(\nu) = \begin{bmatrix} X_u + X_{|u|u}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{uv}|u| + Y_{uv|u|}|v||u| & 0 & 0 & 0 & Y_{ur}|u| + Y_{ur|r|}|r||u| \\ 0 & 0 & Z_{uw}|u| + Z_{uw|w|}|w||u| & 0 & Z_{uq}|u| + Z_{uq|q|}|q||u| & 0 \\ 0 & 0 & 0 & K_{up}|u| + K_{up|u|}|p||u| & 0 & 0 \\ 0 & 0 & M_{uw}|u| + M_{uw|u|}|w||u| & 0 & M_{uq}|u| + M_{uq|q|}|q||u| & 0 \\ 0 & N_{uv|u|} + N_{uv|v|}|v||u| & 0 & 0 & 0 & N_{ur}|u| + N_{ur|r|}|r||u| \end{bmatrix} \tag{3.2.9}$$

Restoring forces:

$$g(\eta) = \begin{bmatrix} (W-B)\sin\theta \\ (W-B)\cos\theta\sin\phi \\ (W-B)\cos\theta\cos\phi \\ -(y_gW - y_bB)\cos\theta\cos\phi + (z_gW - z_bB)\cos\theta\sin\phi \\ (z_gW - z_bB)\sin\theta + (x_gW - x_bB)\cos\theta\cos\phi \\ -(x_gW - x_bB)\cos\theta\sin\phi - (y_gW - y_bB)\sin\theta \end{bmatrix} \tag{3.2.10}$$

Transformation matrix:

$$J(\eta) = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s\phi + c_\psi c_\phi s_\theta & 0 & 0 & 0 \\ s_\psi c_\theta & c_\psi c_\phi + s_\phi s_\theta s_\psi & -c_\psi s_\phi + s_\theta s_\phi c_\phi & 0 & 0 & 0 \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & 0 & 0 & 0 & c_\phi & -s_\phi \\ 0 & 0 & 0 & 0 & s_\phi/c_\theta & c_\phi/c_\theta \end{bmatrix} \tag{3.2.11}$$

where

$$\begin{aligned} s_\theta &= \sin\theta \\ c_\theta &= \cos\theta \\ s_\phi &= \sin\phi \\ c_\phi &= \cos\phi \\ s_\psi &= \sin\psi \\ c_\psi &= \cos\psi \\ t_\theta &= \tan\theta \end{aligned} \tag{3.2.12}$$

The vector η denotes the position and orientation given in the earth fixed coordinate system NED. This frame is a flat-earth approximation and is considered inertial in this model. Such an approximation is considered to provide sufficient accuracy for this sort of simulations, as the effects of the earth rotation and curvature are minimal compared to the other forces and moments. The vector ν denotes the linear and angular velocity vectors which are decomposed in the hull-fixed reference frame BODY (see 3.2.3). Equation 3.2.1 represents the transformation between BODY and NED and the matrix J is simply the transformation matrix obtained from the AUV euler angles given in NED. Equation 3.2.2 represent the torque/force and acceleration balance of the system decomposed in BODY, derived from newtons laws of motion. Starting on the left side, the matrix M (3.2.4) denotes the mass and inertia (including added mass) of the vessel. The term $C(\nu)\nu$ (3.2.5) denotes the nonlinear coriolis and centripetal forces (including added mass) acting on the vessel. $D(\nu)\nu$ (3.2.9) denotes the system damping matrix. Worth noticing about the system damping is the lack of a constant damping term. In fact, all damping terms are multiplied by the surge speed which leads to zero damping once the surge speed is zero. This indicates that the model is designed to be valid only around the nominal surge speed. The last term on the left side is $g(\eta)$ (3.2.10), which represent the forces due to gravity. τ is the complete control input vector consisting of forces applied by the rudders and the propeller. This is the only term in this equation which may be controlled during operation. The vector g_0 may be used to represent pre-trimming of the vessel using ballast tanks or similar systems. For the case of simulating the HUGIN AUV in a collision avoidance context, this is not very relevant and the term was therefore excluded. The last term w represents the external forces acting on the vessel, which for an underwater vessel consists only of varying currents. A constant current will not represent any acceleration to the system in steady state, and equation 3.2.2 is therefore not affected. By assuming the current to be constant or very slow varying, this term was also neglected. Constant linear current may instead be simulated by adding a term $\dot{\eta}_c$ to equation 3.2.1 consisting of the linear velocity of the current given in NED. The resulting equations used for modelling the HUGIN AUV are shown in 3.2.13 and 3.2.14. For further details about equations 3.2.1 and 3.2.2 and modelling of marine vessels, the reader is referred to Fossen (2002) chapters 2-4.

$$\dot{\eta} = J(\eta)\nu + \dot{\eta}_c \tag{3.2.13}$$

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau \tag{3.2.14}$$

3.2.2 Propeller and rudder

As seen in figure 2.1, the HUGIN AUVs have 4 rudders and 1 propeller. Two of the rudders are for horizontal control (yaw) and two of the rudders are for vertical control (pitch). All of the 4 rudders may be deflected independently up to a given maximum angle in each direction. Roll control may be applied by deflecting two adjacent rudders differently. Rudder actuation is also limited by the maximum rate of the rudder-servos. The propeller input is desired propeller rpm.

Propeller modelling was performed using the quasi-steady thrust and torque approach (Fossen 2002, chap 12.2.1). In this approach the surge-thrust T and the roll-torque Q was modelled as shown in equations 3.2.15 and 3.2.16. Here u is the surge speed, n is the propeller angular rate and $T_{n|n|}$, $T_{|n|u}$, $Q_{n|n|}$ and $Q_{|n|u}$ are propeller parameters.

$$T = T_{n|n|}n|n| - T_{|n|u}|n|u \tag{3.2.15}$$

$$Q = Q_{n|n|}n|n| - Q_{|n|u}|n|u \tag{3.2.16}$$

The force and torque vector caused by the propeller (τ_{pr}) may then be derived as:

$$\tau_{pr} = \begin{bmatrix} T \\ 0 \\ 0 \\ Q \\ 0 \\ 0 \end{bmatrix} \tag{3.2.17}$$

The forces and torques caused by the rudders was modelled by a quadratic method, derived from the hugin team. The vector δ consisting of the four rudder angles is denoted as:

$$\delta = \begin{bmatrix} \delta_{top} \\ \delta_{bottom} \\ \delta_{port} \\ \delta_{starboard} \end{bmatrix} \tag{3.2.18}$$

The rudder force/torque lift vector τ_{rl} may be estimated by equation 3.2.19. The matrix B is defined by equation 3.2.20 where $Y_{\delta u^2}$ and $Z_{\delta u^2}$ are rudder parameters and l_x, l_y and l_z defines the position of the rudders in BODY coordinates.

$$\tau_{rl} = B_l \delta u^2 \tag{3.2.19}$$

$$B_l = \begin{bmatrix} 0 & 0 & 0 & 0 \\ Y_{\delta u^2} & Y_{\delta u^2} & 0 & 0 \\ 0 & 0 & Z_{\delta u^2} & Z_{\delta u^2} \\ Y_{\delta u^2} l_z & -Y_{\delta u^2} l_z & -Z_{\delta u^2} l_y & Z_{\delta u^2} l_y \\ 0 & 0 & Z_{\delta u^2} l_x & Z_{\delta u^2} l_x \\ -Y_{\delta u^2} l_x & Y_{\delta u^2} l_x & 0 & 0 \end{bmatrix} \tag{3.2.20}$$

When the rudders are used for actuation, they also contribute to increased drag on the AUV. This force τ_{rd} may be estimated by equation 3.2.21 where $X_{\delta^2 u^2}$ is a negative rudder parameter.

$$\tau_{rd} = \begin{bmatrix} X_{\delta^2 u^2} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \delta^T \delta u^2 \tag{3.2.21}$$

The complete input vector of forces and torques (τ) may then be calculated as:

$$\tau = \tau_{pr} + \tau_{rl} + \tau_{rd} \tag{3.2.22}$$

The physical limitations of the rudders (max angle and max angle rate) were implemented by a saturation and a rate limiter block on the rudder input.

3.2.3 Matlab/Simulink implementation

Implementation in Simulink was performed using an embedded matlab block. The M-code is simply the forming of the matrices in 3.2.13 and 3.2.14 and the calculation of the same equations. As described in Smith & Ghidella (2004), this method of incorporating M-code into Simulink results in comparable running times to the s-function method, when debugging mode is disabled. The embedded matlab block was chosen for its simplicity in implementation, and the simulink diagram is shown in Appendix B.1. The simulink file *hugin1000.mdl*, including the embedded matlab code is included on the attached CD.

Since the model parameters of the HUGIN AUV are confidential, they are not included in the official part of the report. Interested readers must contact FFI or Kongsberg Maritime for access to these data, as the author is not authorized to distribute them further. The complete model with all parameters is included on the CD, attached to the restricted version of the project report which is available only to examiners and guidance supervisors.

3.3 AUV Model Simulations and discussion

It is difficult to confirm the correctness of the implemented HUGIN model as the dynamics are very complex. Also, this is considered out of the scope of this project. Still, the model was run trough a series of tests to get an indication of stability properties and to investigate maximum rates in pitch and yaw.

3.3.1 Test 1: Zero rudder input

In this test, the complete model described in section 3.2.1 was simulated with zero rudder inputs and 2.65 revs/sec propeller speed (nominal) with initial speed 0. A plot of the 6 relevant states are shown in figure 3.3. As seen in the figure, the roll angle started deviating as soon as the simulation began. This is because of roll torque caused by the propeller and the lack of a constant damping term in roll. As the surge speed increased, the roll damping also increased, and gravity forces decreased the roll angle back to an offset of about 0.0175 radians (1 degree). The surge speed gradually increased and settled at the nominal speed (2.05 m/s) after about 50 seconds. The yaw and pitch angles and the sway and heave motions experienced rapid deviations at the very beginning of the simulation, although no rudder torque was present. The only explanation for this must be the lack of constant damping terms for these states, as mentioned in section 3.2.1. This leads to virtually no damping when the surge speed is close to zero. To sum this up, the vessel appears to be relatively stable at nominal speed with no rudder inputs, as all states converge to constant values once the surge speed approaches the nominal speed. However, the simulation indicates that the model is not valid around zero surge speed. This should not be a problem for the use in this project, as all simulations will be performed at nominal speed.

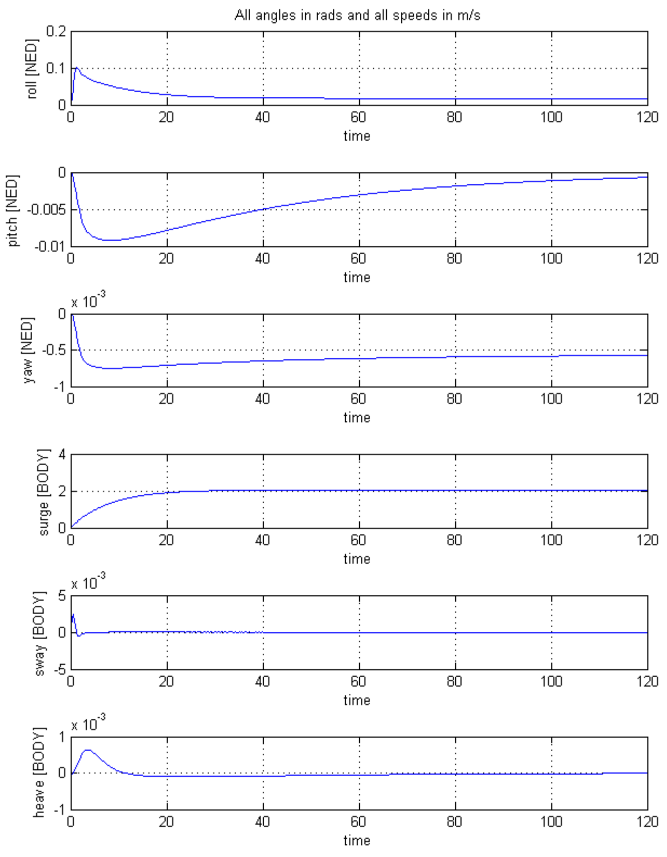


Figure 3.2: Hugin model test 1 - Zero rudder input

3.3.2 Test 2: Maximum yaw rate

In this test, the model was simulated with top and bottom rudder set to the saturation point (12.5 degrees) and the propeller at nominal turn rate, in order to find the maximum yaw rate and investigate the dynamics in this state. In this test the surge speed was initialized at the nominal speed. The result is shown in figure 3.3.

As seen in the figure, the yaw rate quickly stabilized at approximately 0.095 rad/s (approx 6 deg/s) which correspond well with the given maximum yaw rate. It is also noticed that the motion in sway is closely connected with the yaw rate. As the yaw rate increased, so did the sway motion. This was expected since the yaw rate causes the vessel to 'side slip' through the water. At max yaw rate, the sway motion stabilized at approximately 0.25 m/s. This causes the vessel to obtain a course-through-water (CTW) angle at about 8 degrees off the actual heading (yaw angle) during maximum yaw rate turns. The surge speed was slightly decreased because of the increased drag and settled at about 1.85 m/s. This is

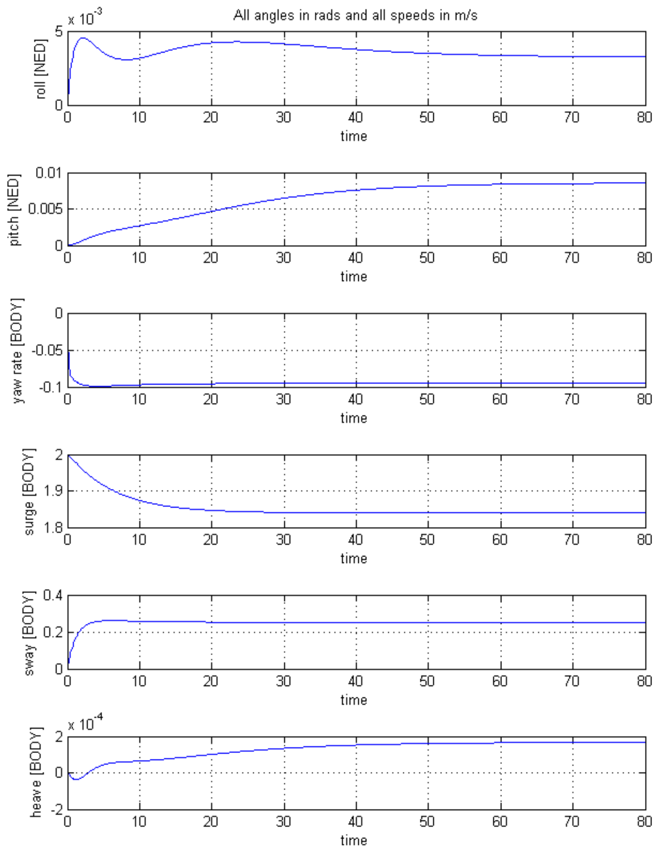


Figure 3.3: Hugin model test 2 - Maximum yaw rate

assumed acceptable and justifies the exclusion of a dedicated speed controller. The roll- and pitch-angles were hardly affected by this maximum turn rate test, which indicates relatively small coupling effects between the states. This should prove benifitial for the decoupled autopilot structure.

3.3.3 Test 3: Maximum pitch rate

The third test was performed to find the maximum pitch rate and the dynamic properties during this maneuver. The simulation was run with port and starboard rudders at saturation levels (12.5 degrees), the propeller at the nominal turn rate and surge initialized at the nominal speed. The result is shown in figure 3.4

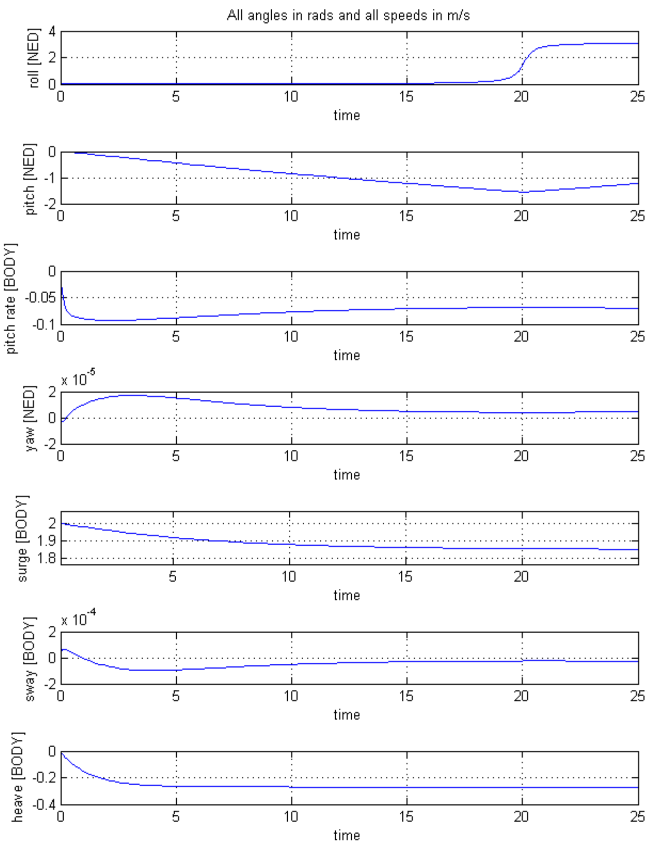


Figure 3.4: Hugin model test 3 - Maximum pitch rate

As seen in the plots, the vessel quickly obtained the maximum pitch rate at approximately 0.096 rad/s (5.5 deg/s). As the pitch angle increased, the forces due to gravity also increased which lead to a gradual reduction in pitch rate. When the pitch angle reached approximately 1.4 radians (80 deg), the roll angle started to deviate and as the pitch angle increased the vessel actually flipped around. This is because the stabilizing effect of the low center of gravity tends to zero at such high pitch angles. The small roll torque applied by the propeller then caused the vessel to flip around. For this reason, the pitch angle operation range has been limited to +-60 degrees. This limitation has also been introduced on the real HUGIN vessel for the same reasons. Within this range, the simulation indicate low coupling into the roll- and yaw states. Similar to the sway motion experienced during the maximum yaw test, the heave motion is closely coupled with the pitch rate, reaching a maximum heave speed of approximately 0.3 m/s during this maneuver

3.3.4 Test 4: All rudders at maximum angle

In this third test, all 4 rudders are set to the saturation point. The simulation was initialized with nominal surge speed and all other states set to zero. The result is shown in figure 3.5.

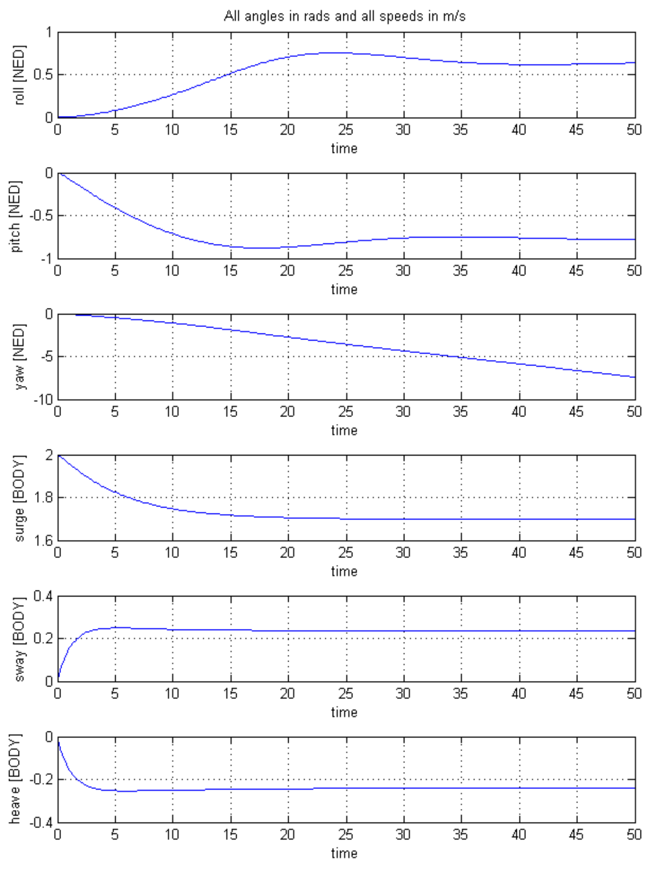


Figure 3.5: Hugin model test 4 - All rudders at max angle

This sort of maneuver represents the most extreme maneuver which may be used for e.g. critical collision avoidance. As seen in the plots, both pitch and yaw quickly started to deviate from the initial condition. At the same time, the roll angle quickly deviated from zero and kept increasing up to an angle of about $43°$. Of course, surveying made under such conditions are not ideal as the surveying sensors are rotated as well. The reason for this large roll deviation is because of simultaneous rates in pitch and yaw which lead to an increasing roll angle. In the model, this effect is implemented in the rotation matrix. The low center of gravity is the only reason the vessel does not flip all the way around. The high roll angle is also the reason the pitch angle never reaches the instability point (ref test 3). Instead the vessel stabilizes in an upwards helix motion.

Chapter 4

Sonar modelling

The term 'SONAR' is defined by Winder (1975) as 'the method or equipment for determining by underwater sound the presence, location or nature of objects in the sea.' It is an acronym for 'Sound Navigation and Ranging'. Active sonar, which is the type discussed in this paper, involves the transmission of an acoustic signal which, when reflected from a target, provides the sonar receiver with a basis for detection and range-estimation. To measure the distance to an object, the time from transmission of a pulse to reception is measured and converted into a range by estimating the speed of sound through water. To measure the bearing, several hydrophones are used to measures the relative arrival time to each. Another method is to use an array of hydrophones and measure the relative amplitude in beams formed through a process called beam forming. The target signal (if present) together with noise is then passed through various forms of signal processing. It is then processed by some form of decision algorithm that determines how the signal is to be interpreted. For further details about sonars and sonar signal processing, the reader is referred to Knight et al. (1981).

4.1 HUGIN sonar configuration

The HUGIN AUV discussed in this paper is assumed equipped with the following sonars (see fig 4.1):

- **Vertical Altimeter** - Single sonar beam pointing straight down.
- **Forward Altimeter** - Single sonar beam pointing down and forward.
- **Starboard Side Scan Sonar** - Wide sector sonar beam primarily used for imaging. Directed $90°$ starboard in the horizontal plane (sector: $0.5°$) and covers a sector from $10°$ to $80°$ (measured from vertical down) in the vertical plane. For ranging, this sonar returns the shortest range within the entire coverage sector.
- **Port Side Scan Sonar** - Identical to the Starboard Side Scan Sonar, only directed port.
- **Forward Looking Sonar (FLS)** - Array of 45 sonar beams covering a sector of $45°$, each covering $1°$ x $15°$..

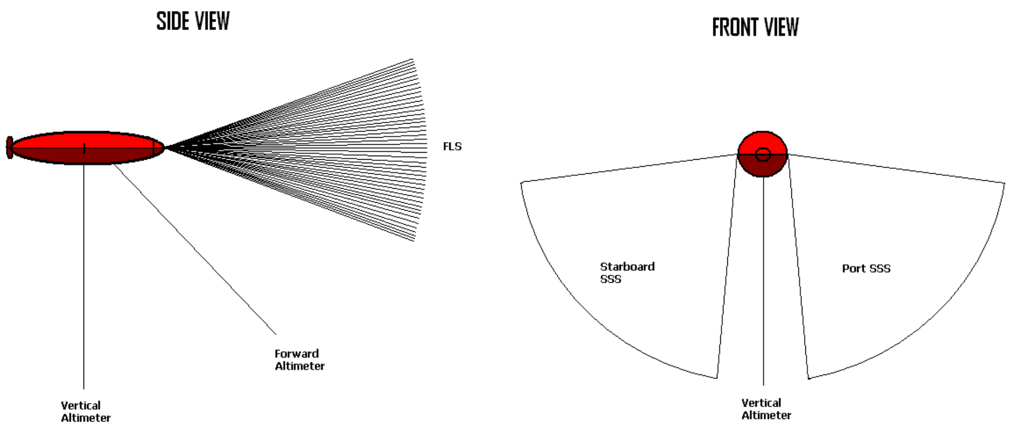


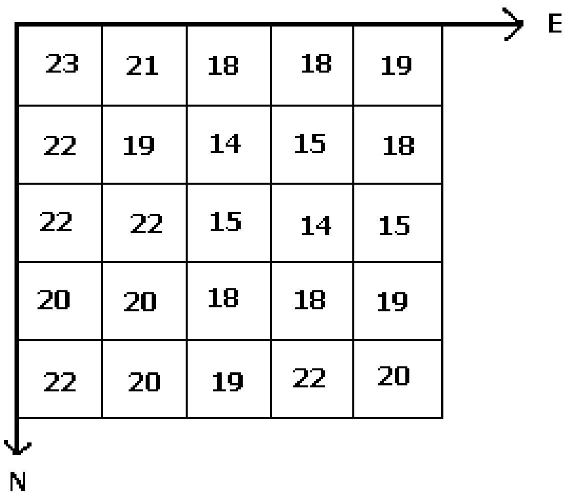
Figure 4.1: Sonar configuration (with vertically mounted FLS)

Sonar	**Orientation** (Euler angles in BODY)	**Range**
Vertical Altimeter	[0 90° 0]	50m
Forward Altimeter	[0 45° 0]	50m
Starboard SSS	[0 (10° → 80°) −90°]	50m
Port SSS	[0 (10° → 80°) 90°]	50m
FLS (vertical configuration)	[0 (−22.5° : 1° : 22.5°) (−7.5° → 7.5°)]	50m

Table 4.1: Sonar configuration

4.2 Digital Terrain Model (DTM)

The simulated environment is represented by a Digital Terrain Model (DTM). The DTM consists of 2 DTM matrices; one containing depths for square sections in the North and East plane as shown in figure 4.2, and the other containing depths of the surface ice in the same area. The depth within each sector is considered equal, and the resolution is determined by how large area each square represents in the real world. A visualization of a DTM (without ice) derived by HUGIN from the Breiangen area in the Oslo fjord, is shown in figure 4.3. This DTM contains 300 x 300 depths and the resolution is 10 meters, so each number in the matrix covers an area of 10m x 10m of the real world. This resolution is used throughout the work on this project but may easily be changed to any value. The matlab script for visualizing DTM matrices (*mapmaker.m*) is included on the CD.



23	21	18	18	19
22	19	14	15	18
22	22	15	14	15
20	20	18	18	19
22	20	19	22	20

Figure 4.2: An example DTM matrix

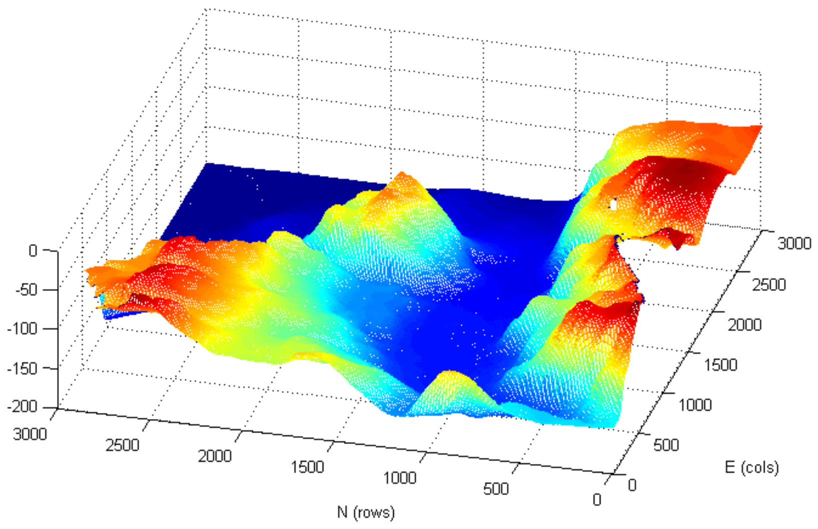


Figure 4.3: A visualization of a DTM for the Breiangen area.

4.3 Developing the sonar simulators

Each sonar range is calculated by iterating along the sonar direction until intersection with the bottom or the ice, represented by the DTM matrices, or until the maximum range is reached. This calculation is performed for each time-step during simulations and must take into consideration the vessel position and attitude as well as the DTM values in the particular area. The single sonar ranges may be calculated in the same way, and this is covered in the first section. The calculation of the side scan sonars and the FLS are somewhat more complicated and are covered in two consecutive sections.

4.3.1 Single sonar beam

In order to find the point of intersection between the bottom profile (or ice) and the sonar beam, the heights of the iteration points must be compared with the bottom height and ice depths. For this reason, a new coordinate frame is introduced; the SONAR frame. This frame has it's origin at the sonar position (given in BODY coordinates) and is rotated with respect to the BODY frame according to the orientations given in table 4.1. Each iteration point P_i is simply given as $[m_i$ 0 0] in SONAR coordinates where m_i is the iteration variable. Starting with $m_i = 0$, the sonar beam is then checked with the desired resolution (δm) by increasing m_i with δm in each step. The iteration points must then be transformed to NED coordinates in order to compare the height to the DTM values in the particular DTM section. Denoting the iteration point in sonar coordinates as p_i^s, the sonar position in BODY coordinates as p_s^b and the rotation matrix from SONAR to BODY coordinates as R_s^b. The position of the iteration point with respect to the BODY frame may be then be found by equation 4.3.1. This may be rewritten into 4.3.2 which is the standard form of homogen transformation used widely within robotics. The homogen transformation matrix (in this case: A_s^b) performs both rotation and translation in one single operation. For further details about homogen transformations, the reader is referred to Sciavicco & Siciliano (2000).

$$\begin{bmatrix} p_i^b \\ 1 \end{bmatrix} = \begin{bmatrix} R_s^b & p_s^b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_i^s \\ 1 \end{bmatrix} \tag{4.3.1}$$

$$P_i^b = A_s^b P_i^s \quad \text{where} \quad A_s^b = \begin{bmatrix} R_s^b & p_s^b \\ 0 & 1 \end{bmatrix}, P_i^b = \begin{bmatrix} p_i^b \\ 1 \end{bmatrix}, P_i^a = \begin{bmatrix} p_i^s \\ 1 \end{bmatrix} \tag{4.3.2}$$

The iteration point p_i^s may be transformed directly into NED coordinates (p_i^n) using the same principle, as shown in equation 4.3.3

$$P_i^n = A_b^n P_i^b = A_s^n P_i^s \quad \text{where} \quad A_s^n = A_b^n A_s^b, \quad P_i^n = \begin{bmatrix} p_i^n \\ 1 \end{bmatrix} \tag{4.3.3}$$

In this fashion, the iteration points may be transformed from the SONAR frame to the NED frame in one single operation and the height of the iteration point may be directly compared with the DTM matrices. The pseudo code for the single sonar beam simulator is shown in table 4.2. This simulation code may be used for each of the single beam sonars, simply by adjusting the orientation and position (in BODY coordinates) in the transformation matrix for each sonar. The matlab source code for calculating single sonar ranges (*singleSonarRange.m*) included on the CD.

```
1. Initialize; m=0
2. Set $P_i^s$=[m 0 0 1]', $P_i^n = A_s^n P_i^s$=[x y z 1]
3. Find correct section in DTM using x and y coordinates of $P_i^n$
4. IF z coordinate is below bottom or above ice $\Rightarrow$ FINISHED, range = m
5. Update range: m=m+$\delta m$
6. IF m > max range $\Rightarrow$ FINISHED, range = m
7. Go back to 2
```

Table 4.2: Single sonar beam simulator pseudo code

4.3.2 Forward Looking Sonar (FLS)

The Forward Looking Sonar (FLS) consists of 45 separate sonar beams which each have to be calculated separately. However, each sonar beam has a horizontal (in body cord.) resolution of 15 degrees. This means that the sonar cannot separate echoes within this sector. To add this effect in the simulator, each beam range is calculated by iterating along 3 lines within the beam sector as shown in figure 4.4 and using the shortest range. This results

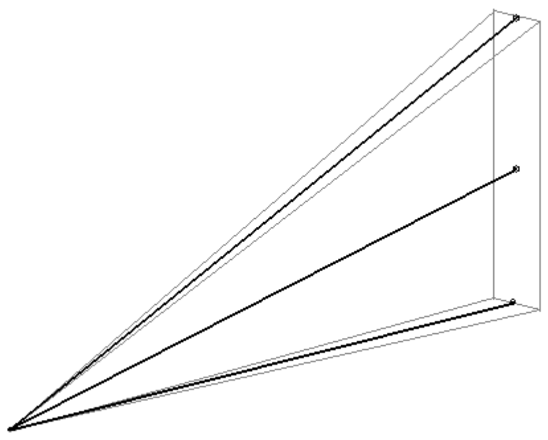


Figure 4.4: Iterating 3 lines within each FLS beam

in the range calculation of 45 x 3 = 135 lines in each time step to calculate the complete set of FLS ranges. The ranges are calculated by introducing two new coordinate frames: SONAR BEAM frame and SONAR LINE frame. The SONAR BEAM frames represents each of the 45 sonar beam sectors. The SONAR LINE frames represent each of the three lines within each beam. Both these frames have origins equal to the SONAR frame, hence only rotation matrices are necessary in order to transform the iteration points to the SONAR frame. The complete set of transformations necessary to transform one iteration point to the

NED frame is given in table 4.3. In this table the following notation is used: R_x^y denotes the rotation matrix from X frame to Y frame, A_x^z denotes the homogen transformation matrix from X frame to Z frame, SONAR LINE = sl, SONAR BEAM = sb, SONAR = s, BODY = b and NED = n. Using the principle of homogen transformation, each transformation may be performed in one single operation as shown in equation 4.3.4, where p_i^{sl} is the iteration point in SONAR LINE coordinates and p_i^n is the same point in NED coordinates.

SONAR LINE frame:	$\mathbf{P_i^{sl}}$=[m 0 0 1]
$\Downarrow (R_{sl}^{sb})$	(rotation)
SONAR BEAM frame:	$\mathbf{P_i^{sb}}$
$\Downarrow (R_{sb}^{s})$	(rotation)
SONAR frame:	$\mathbf{P_i^{s}}$
$\Downarrow (A_s^b)$	(rotation + translation)
BODY frame:	$\mathbf{P_i^{b}}$
$\Downarrow (A_b^n)$	(rotation + translation)
NED frame:	$\mathbf{P_i^{n}}$

Table 4.3: Transforming one FLS iteration point to NED frame

$$\begin{bmatrix} p_i^n \\ 1 \end{bmatrix} = A_{sl}^n \begin{bmatrix} p_i^{sl} \\ 1 \end{bmatrix} = A_b^n A_{sl}^b \begin{bmatrix} p_i^n \\ 1 \end{bmatrix} = \begin{bmatrix} R_b^n & p_b^n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_s^b R_{sb}^s R_{sl}^{sb} & p_s^b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_i^{sl} \\ 1 \end{bmatrix} \tag{4.3.4}$$

The pseudo code for the FLS simulator is given in table 4.4 The implemented matlab code *SonarRanges.m* is included on the CD.

```
1. For each of the 45 beams, form $A_{sb}^n$
2. For each of the 3 lines in each beam form $A_{sl}^n$, calculate range using
   single sonar script 4.2, return smallest value of the three.
```

Table 4.4: FLS simulator pseudo code

4.3.3 Side scan sonar

The two side scan sonars are identical and are calculated by the same method. Each of the sonars have a vertical resolution of 70 degrees. The sonars return the shortest range within this sector in the same way as each of the FLS beams. In order to incorporate this effect in the simulator, each of the ranges are calculated by iterating along 10 lines within the beam sectors of which the shortest range is returned. The calculation of each of the side scan sonars are performed in the same way as with each of the FLS beams, by introducing SONAR LINE frames for each of the lines to be checked. The set of transformation necessary to transform one iteraion point to the NED frame is given in table 4.5, where the same notation as with the FLS is used. The implemented matlab code for calculating the side scan

SONAR LINE frame:	$\mathbf{P_i^{sl}}$=[m 0 0 1]
$\Downarrow (R_{sl}^{s})$	(rotation)
SONAR frame:	$\mathbf{P_i^{s}}$
$\Downarrow (A_{s}^{b})$	(rotation + translation)
BODY frame:	$\mathbf{P_i^{b}}$
$\Downarrow (A_{b}^{n})$	(rotation + translation)
NED frame:	$\mathbf{P_i^{n}}$

Table 4.5: Transforming one SSS iteration point to NED frame

sonar ranges *SideScan.m* is included on the CD.

4.3.4 Simulink implementation

As the sonar simulator performs a great number of calculations in each time step, the simulation speed was significantly reduced after implementing this functionality. One of the reasons for this is the large amount of data that is passed around when reading from the DTM matrices. In order to speed up the simulations, a small modification was made. Instead of reading DTM data from the complete DTM matrices, the data was obtained from a small section of them, covering an area of 50m X 50m around the current position of the AUV. This must be done for both DTMs, and the implemented matlab functions for this task (*makeSDTM.m* and *makeSICE.m*) are included on the CD. The complete script for simulating all sonars *SonarSimulator.m* is also included on the CD. This script was included in the simulink model using an embedded matlab block.

4.4 Sonar simulations and comments

Single sonar beam simulator

The single sonar beam simulator was tested by letting the model run along the surface (depth=0m) through an example landscape. The depth was then measured by the vertical altimeter. The measured depth is plotted against the actual depth in figure 4.5. The sonar resolution was set to 0.1 meters during the run.

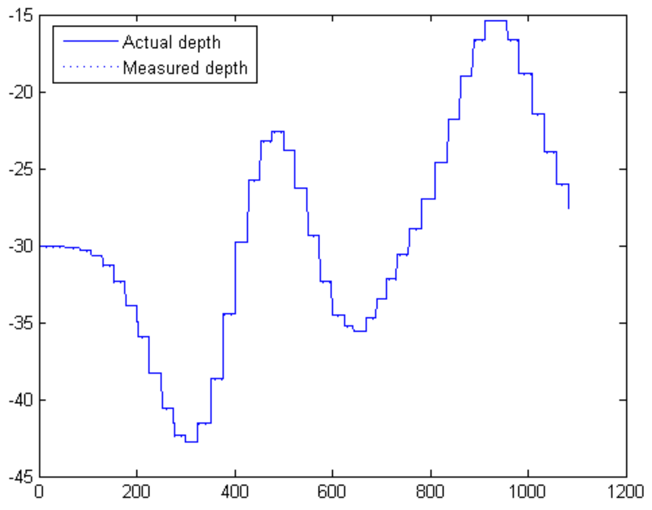


Figure 4.5: Testing the single sonar simulator

Figure 4.5 indicates that the single sonar simulator is accurate. The measured depth is almost identical to the actual dept. A closer analysis of the plot shows that the single sonar measurement is at max 0.1 meters away from the actual depth. This is because of the chosen sonar resolution (δm) which is set to 0.1 meters. Decreasing this resolution would result in even more accurate sonar ranges. This would also decrease the simulation speed, as the simulator has to perform more iterations in order to calculate the distance. For this reason, 0.1 meters is chosen as the simulated resolution as it provides enough accuracy for the system to function properly. This is also close to the resoultion one could expect from a physical sonar.

FLS simulator

One of the tests performed with the FLS simulator was to point the AUV model straight towards a vertical wall at a distance of 40 meters with a horizontal FLS configuration. A visualization of the resulting FLS measurements is shown in figure 4.6.

The figure indicates that the FLS simulator calculates all of the sonar ranges correctly. Again, the small variations is assumed to be a result of the chosen sonar resolution. Further tests using higher resolutions confirms this. The correctness of the FLS simulator was also confirmed in a number of additional tests.

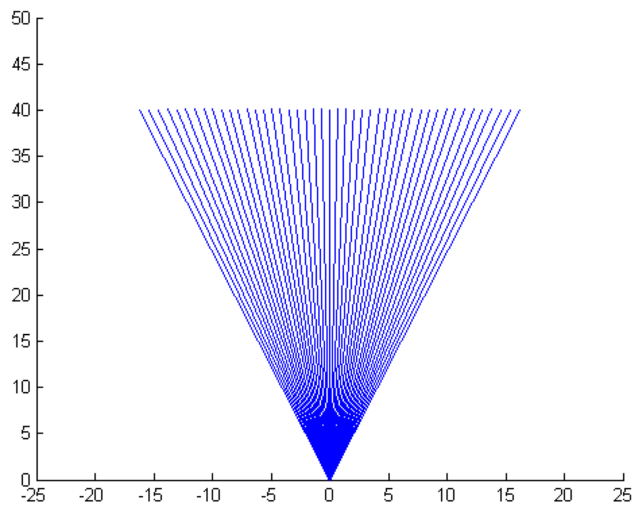


Figure 4.6: Testing the FLS simulator

Side scan sonars

The side scan sonar simulator is not easily tested because of the wide sector and low resolution. Although some simple tests were performed indicating correct implementation. In addition, the calculations rely on the single sonar beam implementation which has been proven correct. For these reasons, the side scan sonar implementation is assumed to be correct. Later simulations using the complete system has confirmed this assumption.

Chapter 5

Autopilot design

In order to simulate the complete system, an autopilot needed to be designed for the AUV model, similar to the one used on the actual HUGIN. For this reason a number of different autopilot versions were developed, specifically designed to work in coherence with the rest of the system (see figure 2.8) and a proposed Collision Avoidance System. The Autopilot should accept input in the form of desired angles in roll, pitch and yaw to fit into this system. There are many choices for the design of an autopilot for AUVs, and this has been addressed by a large number of authors. Some useful references are Healey & Lienard (1993), Fryxell et al. (1996), Pascoal et al. (1997) and Leonard (1997). As with almost any physical control design, the principle choice is whether the autopilot is to be based on nonlinear or linear design. While a nonlinear design has the potential of global stability (Khalil 2000) and to handle rigorous nonlinearities in the physical system, a linear design is often preferred because of its simplicity in tuning and good practical performance (Chen 1999). This chapter describes the development and implementation of several linear designs. In section 5.5 a nonlinear feedback linearization scheme is derived and applied to the AUV model. It should be noted that autopilot design is not considered the main core of this project work.

Although the roll dynamics of the HUGIN are inherently stable, as the center of gravity is below the center of buoyancy, additional roll control was applied in the autopilots in order to maintain a minimal roll angle during all maneuvers. This was to ensure maximum utilization of the payload sensors and to minimize coupling effects in pitch and yaw. Surge speed control was not implemented, as the vessel surge speed is considered sufficiently uniform using a fixed propeller revolution. In addition, stable surge speed is not considered critical for the overall operation.

5.1 Linearized and reduced vessel model

For the purpose of a linear autopilot design, a linear model describing the dominating dynamics of roll, pitch and yaw was needed. For this reason, the nonlinear model in 3.2.13 and 3.2.14 was reduced by assuming sway and heave to be small and surge to be the nearly constant ($u_0 = 2,0474$ m/s). The simulations performed in chapter 3 confirms this assumption. In addition, including these states in the autopilot model has no purpose as the rudders may only apply torques in roll, pitch and yaw. This resulted in the following reduced (but still nonlinear) model:

$$M\dot{\begin{bmatrix}p\\q\\r\end{bmatrix}} + C\begin{bmatrix}p\\q\\r\end{bmatrix} + D\begin{bmatrix}p\\q\\r\end{bmatrix} + \begin{bmatrix}0\\Z_gW\sin(\theta)\\0\end{bmatrix} = \begin{bmatrix}\tau_p\\\tau_q\\\tau_r\end{bmatrix} \tag{5.1.1}$$

where:

$$M = \begin{bmatrix}I_x & 0 & 0\\0 & I_y - M_{\dot{q}} & 0\\0 & 0 & I_z - N_{\dot{r}}\end{bmatrix} \tag{5.1.2}$$

$$C = C_{RB} + C_A \tag{5.1.3}$$

$$C_{RB} = \begin{bmatrix}0 & I_zr & -I_yq\\-I_zr & 0 & I_xp\\I_yq & -I_xp & 0\end{bmatrix} \tag{5.1.4}$$

$$C_A = \begin{bmatrix}0 & -(Y_{\dot{r}}v + N_{\dot{r}r}) & Z_{\dot{q}}w + M_{\dot{q}}q\\Y_{\dot{r}}v + N_{\dot{r}}r & 0 & -K_pp\\-(Z_{\dot{q}}w + M_{\dot{q}})q & K_pp & 0\end{bmatrix} \tag{5.1.5}$$

$$D = \begin{bmatrix}K_{up}|u_0| & 0 & 0\\0 & M_{uq}|u_0| & 0\\0 & 0 & N_{ur}|u_0|\end{bmatrix} \tag{5.1.6}$$

Linearizing this model about zero roll and pitch, yields the linear differential equations in 5.1.7, 5.1.8 and 5.1.9. As seen from these equations, coriolis effect has been eliminated by the linearization, resulting in three linear decoupled differential equations.

$$\dot{p} = -\frac{K_{up}|u_0|p}{I_x} + \frac{\tau_p}{I_x} \tag{5.1.7}$$

$$\dot{q} = -\frac{M_{uq}|u_0|q}{I_y - M_{\dot{q}}} - \frac{Wz_g\theta}{I_y - M_{\dot{q}}} + \frac{\tau_q}{I_y - M_{\dot{q}}} \tag{5.1.8}$$

$$\dot{r} = -\frac{N_{ur}|u_0|r}{I_z - N_{\dot{r}}} + \frac{\tau_r}{I_z - N_{\dot{r}}} \tag{5.1.9}$$

Including the NED states; ϕ, θ and ψ with the linearized relationships given in 5.1.10, the total linearized model may be represented in state space, as shown in equation 5.1.11. As seen in 5.1.11, the linearized model is completely decoupled in roll, pitch and yaw. Hence, the resulting controllers based on this model will not take into account the coupling effects present in the real vessel.

$$\dot{\phi} = p, \quad \dot{\theta} = q, \quad \dot{\psi} = r \tag{5.1.10}$$

$$\begin{bmatrix}\dot{p}\\\dot{q}\\\dot{r}\\\dot{\phi}\\\dot{\theta}\\\dot{\psi}\end{bmatrix} = \begin{bmatrix}-\frac{K_{up}|u_0|}{I_x} & 0 & 0 & 0 & 0 & 0\\0 & -\frac{M_{uq}|u_0|}{I_y - M_{\dot{q}}} & 0 & 0 & -\frac{Wz_g}{I_y - M_{\dot{q}}} & 0\\0 & 0 & -\frac{N_{ur}|u_0|}{I_z - N_{\dot{r}}} & 0 & 0 & 0\\1 & 0 & 0 & 0 & 0 & 0\\0 & 1 & 0 & 0 & 0 & 0\\0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}p\\q\\r\\\phi\\\theta\\\psi\end{bmatrix} + \begin{bmatrix}\frac{1}{I_x} & 0 & 0\\0 & \frac{1}{I_y - M_{\dot{q}}} & 0\\0 & 0 & \frac{1}{I_z - N_{\dot{r}}}\\0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix}\begin{bmatrix}\tau_p\\\tau_q\\\tau_r\end{bmatrix} \tag{5.1.11}$$

5.2 LQ rudder control

This controller was designed to output the 4 different rudder angles directly. For this reason, the linearized model used in designing the feedback gains, had to be extended by including the effect of rudder deflections. Since the relationship between rudder deflections and resulting torque is already linear, this extension was performed by simply including the rudder control matrix (3.2.20) in the linearized model (5.1.11). This resulted in the linear model given in 5.2.1. The rudder drag effect is not included in this equation as it only affects the surge speed.

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{K_{up}|u_0|}{I_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{M_{uq}|u_0|}{I_y - M_{\dot{q}}} & 0 & 0 & -\frac{Wz_g}{I_y - M_{\dot{q}}} & 0 \\ 0 & 0 & -\frac{N_{ur}|u_0|}{I_z - N_{\dot{r}}} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \\ \phi \\ \theta \\ \psi \end{bmatrix} + \begin{bmatrix} \frac{Y_{du^2} l_z}{I_x} & -\frac{Y_{du^2} l_z}{I_x} & -\frac{Z_{du^2} l_y}{I_x} & \frac{Z_{du^2} l_y}{I_x} \\ 0 & 0 & \frac{Z_{du^2} l_x}{I_y - M_{\dot{q}}} & \frac{Z_{du^2} l_x}{I_y - M_{\dot{q}}} \\ -\frac{Y_{du^2} l_x}{I_z - N_{\dot{r}}} & -\frac{Y_{du^2} l_x}{I_z - N_{\dot{r}}} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_{top} \\ \delta_{bottom} \\ \delta_{port} \\ \delta_{starboard} \end{bmatrix} u_0^2 \tag{5.2.1}$$

This state space model is of standard form and may be written as:

$$\dot{x} = Ax + B\delta, \quad \delta = \begin{bmatrix} \delta_{top} & \delta_{bottom} & \delta_{port} & \delta_{starboard} \end{bmatrix}^T \tag{5.2.2}$$

The controller was then designed by applying feedback as shown in 5.2.3, where the input are the desired angles in roll (ϕ_d), pitch (θ_d) and yaw (ψ_d).

$$\delta = -K \begin{bmatrix} p \\ q \\ r \\ \phi - \phi_d \\ \theta - \theta_d \\ \psi - \psi_d \end{bmatrix} \tag{5.2.3}$$

The feedback gain matrix K may be found using different methods of linear feedback, eg. Pole Placement (see Chen 1999, chap 9) or Linear Quadratic (LQ) optimization (see Foss 2004). A Linear Quadratic design was chosen for its intuitive method of tuning. Finding the feedback gains then consisted of solving the optimization problem 5.2.4 and 5.2.5. This was performed using the matlab function *lqr()*.

$$min \quad J(u) = \int_0^\infty x^T Q x + \delta^T R \delta \quad dt \tag{5.2.4}$$

subject to equality constraints (system dynamics):

$$\dot{x} = Ax + B\delta \tag{5.2.5}$$

The input matrix R was set to the identity matrix, as all 4 rudders are equal and their deflection should be weighted equally. Tuning the controllers therefore consisted of setting the state weight matrix Q. All off-diagonal elements were set to zero as the states are completely decoupled. The diagonal elements representing each of the three controlled states (roll,pitch and yaw) determines how much weight the controller should put on each of them. As stated earlier, the roll angle should be kept close to zero at all times in order to maximize payload sensor utilization and to keep coupling effects to a minimum. For this reason the roll state weight was set slightly larger than the pitch- and yaw state weights. The resulting weight matrices used for calculating the gain matrix are shown in (5.2.6).

$$Q = 300 \times \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2.6}$$

The m-file used to calculate the controller gains *LQrudder.m* is included on the restricted CD.

5.2.1 Simulation and comments

In order to test and compare the different autopilot designs, they were all run through the same test. This test consisted of a simultaneous step in pitch and yaw, large enough to cause saturations for all rudders. This is assumed to represent the most extreme case for the autopilot. The test were initialized with all states set to zero, except for the surge speed which was set to the nominal speed (2,05 m/s). At time = 10, the desired yaw was set to 90 degrees ($\pi/2$ rads) and desired pitch to 45 degrees ($\pi/4$ rads). In the simulation plot 5.1, the reference inputs are marked red and the measured state outputs are marked blue.

Because of the large step applied to pitch and yaw at time=10s, all four rudders were saturated for a period of 10 seconds, only delayed by the rudder rate saturations. This resulted in the same behavior as in test 4 (maximum rudder test, figure 3.5) where the roll angle started deviating immediately. Although the roll controller was active during this time, the applied roll-torque was suppressed by the rudder saturations. Only when the roll deviation reached approximately 0.15 radians, the applied roll torque became visible on the rudder outputs. And once the pitch and yaw angle approached the reference angles, the roll controller steadily gained control of the rudders, eventually forcing the roll angle back to the equilibrium point. This behavior is caused by the continuous struggle between the controllers, all trying to apply different torques on the rudders. When the rudders are saturated, the desired roll torque is simply suppressed.

Effectively, this autopilot consists of 3 separate PD controllers for roll,pitch and yaw. In practice however, these controllers are not separate, as the same control inputs are used to control different states (see equation 3.2.20). Although yaw and pitch command are

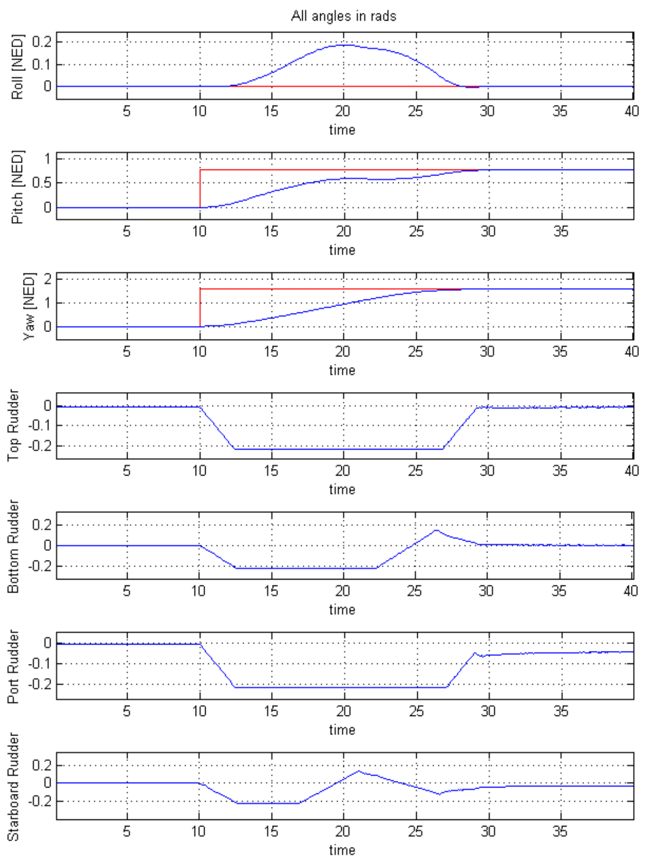


Figure 5.1: Testing the LQ rudder autopilot

separated, the roll control input affect both pitch and yaw control as they use the same rudders to apply torque. Because of the rudder saturations, this causes the roll control to conflict with yaw and pitch control. Say the roll controller attempts to apply roll torque using the rudders. Then it will add a small deflection angle to 2 of the rudders (e.g starboard and top) and subtract a small deflection angle to the other two. If the desired deflection commanded by the pitch and yaw controllers are already well above the saturation points, the small deflections from the roll-controller will not have any effect on the actual rudder outputs as they are already at the saturation points. As a result, no roll torque is applied leading to free roll response. The internal priorisation of the three states may be adjusted to a certain degree by altering the weight matrices proportionally, but is still mostly governed by the current rudder saturations. Also, adjusting the weight matrices to much could degrade the performance and may also cause unnecessary 'stiffness' in the system, causing it to be more sensitive to measurement noise.

The torque forces from the propeller caused a small steady state error on the roll angle (approximately 0.0005 radians). In addition, the gravity forces causes a steady state error on the pitch angle of about 0.03 radians (1.7 degrees). This offset may have been handled by various strategies. One option is including an integrated pitch state, but as the pitch torque suffers from heavy saturations one would have to include some sort of anti-windup strategy as well, in order to avoid windup problems. Another option would be to feedforward the gravity forces and this option would probably yield the best results. However, the offset was considered acceptably small so no strategy was applied. As the torque due to gravity is linearly dependent on the pitch angle, the steady state error experienced should never exceed 0.03 radians.

5.3 LQ torque control with rudder allocation algorithm

In order to better control the internal priorisation of the different states, an alternative autopilot strategy was developed. This autopilot uses the LQ method in a similar way, but instead calculates the desired torque for each of the controlled states. (roll, pitch and yaw). The model derived in 5.1.11 was used to calculate the LQ controller gains, again using the matlab function *lqr()*, and the weight matrices were used to tune the performances separately. For details, see the matlab script *LQtorque.m* which is included on the CD. In this way, the controller outputs the desired torque for each of the states, and not the rudder deflections directly. Internal priorisation of the different states may then be achieved by controlling which of the desired torques are to be prioritized in the rudder deflections. A rudder allocation algorithm was developed for this purpose. This algorithm has the option of prioritizing roll control before pitch- and yaw-control. The algorithm must take into consideration the direction of the applied pitch- and yaw-torque together with the direction of the desired roll torque obtained from the controller. Using this information, the algorithm decides which rudder needs to be relaxed or increased in order to apply the desired roll torque. The algorithm first attempts to apply the desired roll torque without affecting applied pitch and yaw torque, by adjusting corresponding rudders equally (e.g. top rudder $+1°$, bottom rudder $-1°$). If this is not possible due to rudder saturations and the algorithm is set to prioritize roll control, the algorithm decreases applied pitch- and yaw-torque equally by relaxing one of the rudders for each state. The overall strategy/pseudo code is shown in table 5.1.

It is worth noticing that this algorithm does not change the control law in any way, it simply controls the state input saturations by deciding which states are to be prioritized. Assuming the roll angle is kept within a certain area, this would effectively stop the roll controller from ever getting saturated, reducing the overall roll deviation. By keeping the roll angle close to zero, the coupling effects are minimized which is assumed to increase the overall performance of the autopilot, as it is designed on the basis of full decoupling. In addition, variations in roll will degrade the quality of payload sensor measurements and should therefore be kept to a minimum. However, this roll-priorisation will necessarily increase the response time of the roll- and pitch controllers in some cases. In some situations, e.g. a critical collision avoidance maneuver, this is not desirable and for this reason the priorisation may be altered during operation. As a result, the autopilot may operate in two different modes; roll-priorisation mode and pitch- and yaw-priorisation mode, hence it may be regarded as a hybrid system (see Branicky 1998). Item number 6 in table 5.1 is only executed when the autopilot is in roll-priorisation mode. Mode switching is performed by the Collision Avoidance system which is described in chapter 10.

1. Obtain desired torques in roll, pitch and yaw from the LQ controllers.
2. Apply desired yaw torque using top and bottom rudder within saturations
3. Apply desired pitch torque using port and starboard rudder within saturations
4. Calculate 'available rudder deflection' for each of the two pairs (top-bottom and port-starboard). This is the difference between the applied rudder deflection and the saturation limit.
5. Apply desired roll torque within the 'available rudder deflection' by applying deflection differences in the rudder pairs. This operation does not alter the applied pitch- or yaw-torque.
6. IF (roll prioritized)
   Apply remaining desired roll torque (if there is any), reducing pitch- and yaw-torque by equal amounts. This operation takes into consideration the direction of the desired roll torque compared with the direction of the rudder saturations and calculates which rudders are to be relaxed.

Table 5.1: Rudder allocation algorithm

In the linear model (5.1.11) used for autopilot design, the BODY-states p, q and r are in fact equal to the derivatives of the controlled NED states ϕ, θ and psi, although this is not true in the nonlinear model. For implementation of these PD controllers, the measurement/estimate used for the D-parts may be either the BODY states or the derivatives of the NED states. Based on simulation results, the derivatives of the NED states was chosen to be used for the rest of the project. Using the derivatives of the NED states is also assumed to be the better choice, as these measurements include the effects of rotating from BODY to NED. The rudder allocation algorithm (*rudderalloc.m*) is included on the CD. The simulink diagram of this autopilot design is shown in appendix B.2.

5.3.1 Simulation and comments

An identical test was performed using this autopilot, consisting of a step input in pitch and yaw after 10 seconds. The result is shown in figure 5.2 where the reference inputs are marked red and the measured state outputs are marked blue. As the rudder allocation algorithm was set to roll priority, the rudders immediately responded to the increasing roll angle. In fact, the roll angle never exceeded 0.005 radians off the 0 reference input compared to 0.15 radians in the pure LQ rudder strategy. However, this use of the rudders comes at the cost of decreased torque in pitch and yaw, causing a delay of approximately 4 seconds to reach steady state for these states compared to the previous autopilot. Setting the priority to pitch and yaw and running the same test, lead to very similar results as with the LQ rudder controller. The only difference was that the pitch and yaw responses were completely uninterrupted by the roll

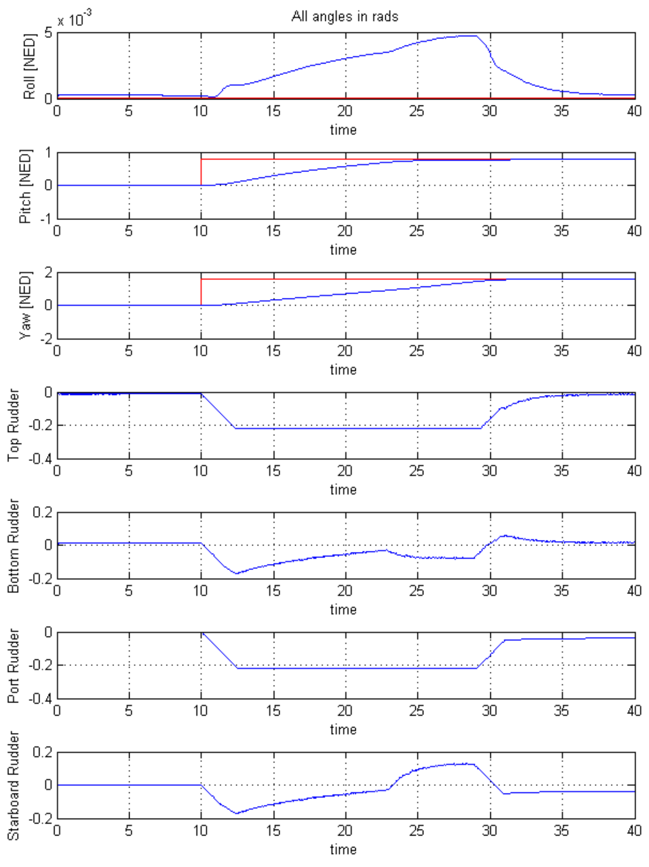


Figure 5.2: Testing the LQ autopilot with rudder allocation algorithm

controller, leading to faster pitch and yaw responses and larger roll deviation. The choice of priority must therefore be made on the basis of user demands and the current situation. E.g in a critical collision avoidance maneuver the roll deviation is irrelevant and the pitch and yaw states should be prioritized in order to achieve the fastest response possible. However, when the vessel is performing measurements sensitive to roll angle deviation, the roll angle should be prioritized as there is no use being in the exact preferred position if the measurements are unusable. For this reason the priority may be altered during operation and controlled by the collision avoidance functionality discussed in chapter 10.

5.4 PID control

Assuming the roll controller not to be saturated, using the rudder allocation algorithm in roll priorisation mode, integral action may be applied in order to eliminate steady state errors (e.g due to propeller torque). For this reason the linear model in equation 5.1.11 was extended by adding a integrated roll state, as shown in equation 5.4.1

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\\ \dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\\ \phi\end{bmatrix} = \begin{bmatrix}-\frac{K_{up}|u_0|}{I_x} & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & -\frac{M_{uq}|u_0|}{I_y-M_{\dot{q}}} & 0 & 0 & -\frac{Wz_g}{I_y-M_{\dot{q}}} & 0 & 0\\ 0 & 0 & -\frac{N_{ur}|u_0|}{I_z-N_{\dot{r}}} & 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0\end{bmatrix} \begin{bmatrix}p\\ q\\ r\\ \phi\\ \theta\\ \psi\\ \int_0^t \phi(\tau)d\tau\end{bmatrix} + \begin{bmatrix}\frac{1}{I_x} & 0 & 0\\ 0 & \frac{1}{I_y-M_{\dot{q}}} & 0\\ 0 & 0 & \frac{1}{I_z-N_{\dot{r}}}\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \begin{bmatrix}\tau_p\\ \tau_q\\ \tau_r\end{bmatrix} \quad (5.4.1)$$

The controller gains were then found by the matlab function *lqr()* and applied to the simulink model as with the PD controllers, in conjunction with the rudder allocation algorithm. For details, see the matlab script (*LQpid.m*) which is included on the CD.

5.4.1 Simulation and comments

The same test was performed on this autopilot, consisting of a step input in pitch and yaw after 10 seconds. The result is shown in figure 5.3 where the reference inputs are marked red and the measured state outputs are marked blue. Effectually, this autopilot forms a PID controller for the roll state. The reason for including the integrated roll state was to minimize slow varying and steady state errors on the roll state. As the plots show, the error is in fact reduced significantly compared to the PD controller in section 5.2. As the roll angle started to deviate after the step inputs, the integrator started to build up and suppress further roll deviation. However, once the maneuver suddenly stopped, all the accumulated roll error caused a deviation in the opposite direction. Still the integrated state controller surpasses the performance of the pure PD controller in terms of roll deviation. The steady state error caused by the propeller was also suppressed by the integral effect. In fact, the simple inclusion of the integrated roll state proved to be surprisingly effective, both in terms of reducing maximum roll deviation and steady state error.

A similar strategy may have been applied to the other states. However, integral action in pitch and yaw would probably suffer from wind-up effects because of the heavy saturations. Anti wind-up strategies could have been applied to reduce this, but integral action was instead excluded in yaw and pitch control as they have shown not to suffer from significant steady state errors.

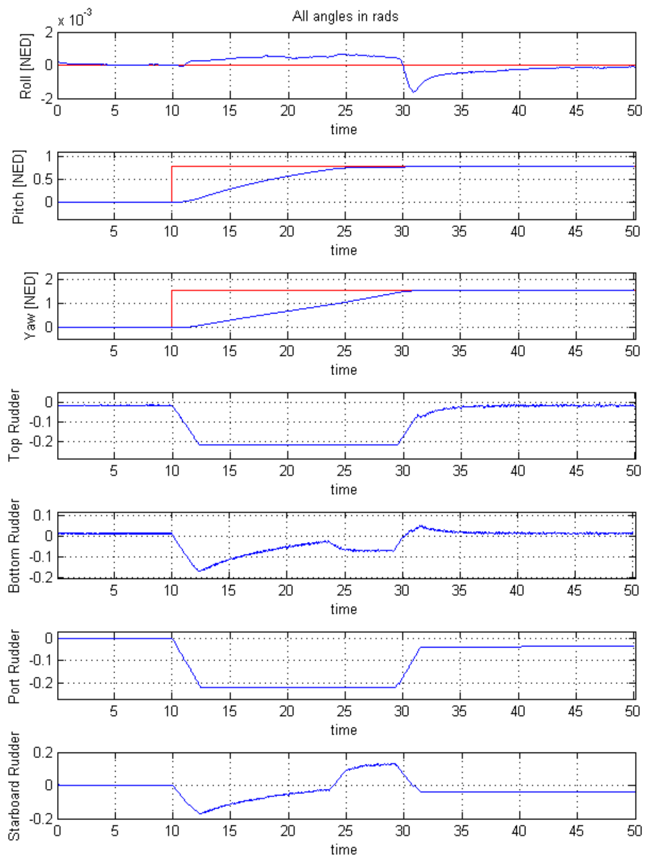


Figure 5.3: Testing the Integrated state autopilot

5.5 Feedback linearization roll control

The various nonlinear and coupling effects present in the vessel (coriolis etc.) causes deviations in roll as they are not taken into consideration in the linear autopilot design. In order to further decrease the errors in roll, a nonlinear feedback linearization scheme was applied. Recall the nonlinear system dynamics equations (3):.

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau \tag{5.5.1}$$

$$\dot{\eta} = J(\eta)\nu \tag{5.5.2}$$

In Fossen (2002), the following derivation is suggested.

$$\ddot{\eta} = a^n \tag{5.5.3}$$

where a^n denotes the NED-frame commanded acceleration. Differentiation of the kinematic equation with respect to time yields:

$$\dot{\nu} = J^{-1}(\eta)[\ddot{\eta} - \dot{J}(\eta)\nu] \tag{5.5.4}$$

Choosing the nonlinear control law

$$\tau = Ma^b + C(\nu)\nu + D(\nu)\nu + g(\eta) \tag{5.5.5}$$

where a^b denotes the commanded acceleration in the BODY frame. Applying this to the system dynamics yields

$$M(\dot{\nu} - a^b) = MJ^{-1}(\eta)[\ddot{\eta} - \dot{J}(\eta)\nu - J(\eta)a^b] \tag{5.5.6}$$

Choosing:

$$a^n = \dot{J}(\eta)\nu + J(\eta)a^b \tag{5.5.7}$$

results in the linear decoupled system

$$J^{-T}(\eta)MJ^{-1}(\eta)(\ddot{\eta} - a^n) = 0, \quad J^{-T}(\eta)MJ^{-1}(\eta) > 0 \tag{5.5.8}$$

From 5.5.7 it can be seen that

$$a^b = J^{-1}(\eta)[a^n - \dot{J}(\eta)\nu] \tag{5.5.9}$$

The commanded acceleration in the NED frame may then be chosen as a linear PD controller

$$a^n = -K_d\dot{\tilde{\eta}} - K_p\tilde{\eta} \tag{5.5.10}$$

Once again the controller gains were found using LQ optimization, but they may have been calculated using any linear feedback design. Although this method could have been applied for all of the controlled states, it was only applied to the roll controller as the pitch and yaw controllers have already proven sufficient performance using pure linear feedback. Cancellation of the propeller torque was also introduced by setting the applied roll torque

$$\tau_{rl}^{roll} = \tau^{roll} - \tau_p^{roll} \tag{5.5.11}$$

This was then applied to the autopilot in combination with the rudder allocation algorithm derived in section 5.3. Of course, such a nonlinear cancellation assumes a perfect hydrodynamical model of the vessel which is unlikely to occur for an AUV due to the complex dynamical environment. The rudder saturations and rate limitations will also degrade the cancellation. However, it is believed that this feedback linearization will reduce the overall roll deviation.

5.5.1 Simulation and comments

Finally the feedback linearization autopilot was run trough the same test, consisting of a step input in pitch and yaw after 10 seconds. The result is shown in figure 5.3 where the reference inputs are marked red and the measured state outputs are marked blue. This control strategy is identical to the PD controller with rudder allocation, except for the nonlinear feedback in roll. In theory, this feedback removes all nonlinearities from the roll dynamics, resulting in a pure linear system. However, the rudder rate saturation causes an inexact cancellation. In addition, model errors and uncertainties would cause additional errors in the nonlinear cancellation. But, as shown in the plots, the strategy decreases the roll deviation

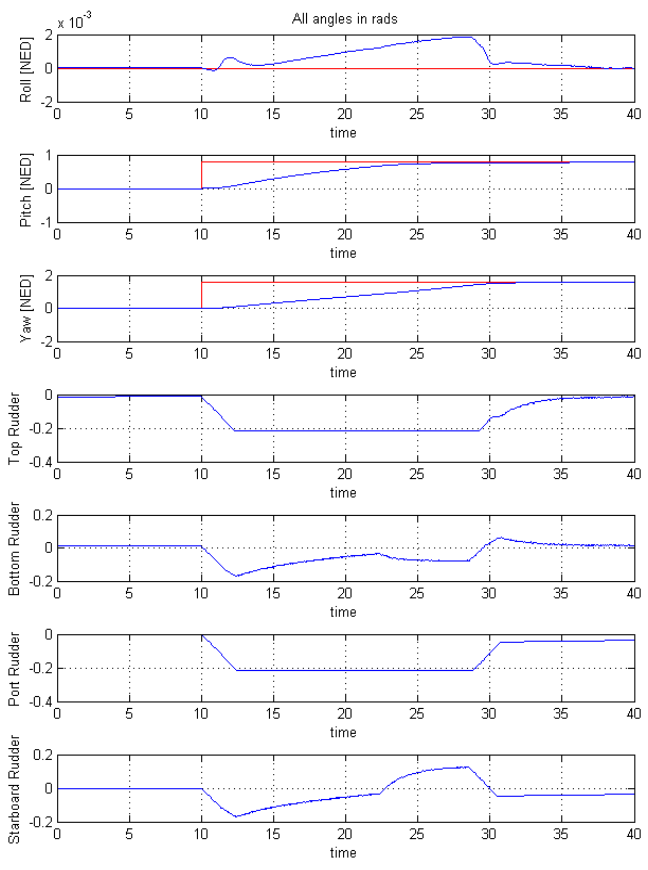


Figure 5.4: Testing the Feedback linearization controller

significantly compared to the pure LQ strategy in the simulations, resulting in a maximum roll deviation of 0.002 radians (0.11 degrees). In addition, the steady state roll-error due to propeller torque is completely removed. This result is based on a perfect model and the only reason the cancellation is not perfect, is the rudder rate saturations. In a practical implementation one can not assume a perfect model which leads to a degradation of the nonlinear cancellations. For this reason a number of test were run with the model used for feedback linearization slightly perturbed. Surprisingly, the controller performed remarkably well, even with relatively large perturbations. A clear disadvantage of this strategy are stability concerns. As the cancellation is never perfect, stability properties are very hard to determine.

5.6 General Autopilot Discussion

The inclusion of the rudder allocation algorithm has proved to give better control of state priorisation in the autopilot design. The two last autopilot options indicate that using this algorithm in combination with additional measures to reduce roll deviation (feedback linearization and integrated roll state), reduces the overall roll error during heavy maneuvering. However, such maneuvers are not considered to occur often during normal operation as most maneuvers are pure pitch or yaw changes. As discussed before, the only situation such a maneuver is likely to happen is during a critical collision avoidance maneuver in which the roll deviation is irrelevant. For this reason the pure LQ torque controller with rudder allocation (section 5.3) has been chosen to be used for the rest of the project. The designs discussed in sections 5.5 and 5.4 may be applied if additional roll control is desired.

The question of stability is an important factor in designing any controller. Deriving nonlinear analytical stability for such a coupled system with 6 degrees of freedom is not straight forward. Such approaches will often use a Lyapunov function and design the controller based on this. The aim of such a controller is to obtain a positive definite lyapunov function $V(x)$ with a negative (semi)definite time derivative $\dot{V}(x)$. Different properties on the $V(x)$, $\dot{V}(x)$ pair leads to different degrees of stability. For details about Lyapunov function based controllers, the reader is referred to Khalil (2000). Autopilots for underwater vehicles based on this approach have been derived by several authors, e.g. Fossen (1991) and Healey & Lienard (1993). Still linear designs are often preferred in practical applications. A reason for this might be that the parameters of nonlinear controllers are often non-intuitive and obtaining desired performance with such controllers may prove to be a challenge. In addition, linear controllers often prove excellent practical performance in AUV autopilot design. The main disadvantage with the linear design is the lack of analytical stability outside the point of linearization. Stability properties for linear controller designs, as discussed in this section, are based on linear stability analysis around the point linearization. Generally this implies that the system poles must have negative real parts. For details see Chen (1999). If the linearized system is stable based on this criteria, this implies asymptotic stability to the nonlinear system around the point of linearization. (Lyapunovs indirect theorem, Khalil (2000) p.139). As all linear controllers in this section are found using the Linear Quadratic method, they are guaranteed stable in the linear systems sense (Foss 2004). This implies that all the autopilots, except for the feedback linearization which requires further analysis, are asymptotically stable around zero pitch and yaw for the entire nonlinear system. Deriving stability properties outside of this point requires additional analysis. Still, extensive simulations indicate that all of the autopilots are indeed stable for the entire operation area (pitch angle between $-60°$ and $+60°$). As autopilot design is not considered to be the main core of this assignment, no further analysis has been derived. However, this is something that should be analyzed further, prior to a practical implementation.

Chapter 6

Line of Sight Waypoint Guidance

The task of the guidance system is to follow the pre-planned horizontal path, consisting of a set of WP's. A line of sight scheme is chosen to simulate the guidance functionality of the HUGIN AUV since the actual guidance functionality is kept confidential. Line Of Sight (LOS) guidance is a simple, yet attractive method for path control. A LOS vector is created from the vessel position to a point ("aiming point") on the current waypoint line, *n* meters ahead of the vessel. This vector defines the desired course which in turn is fed into the autopilot. The concept is illustrated in figure 6.1. A circle-of-acceptance is then created around the next WP and the guidance algorithm switches to the next WP set when the vessel reaches this circle (Fossen 2002).

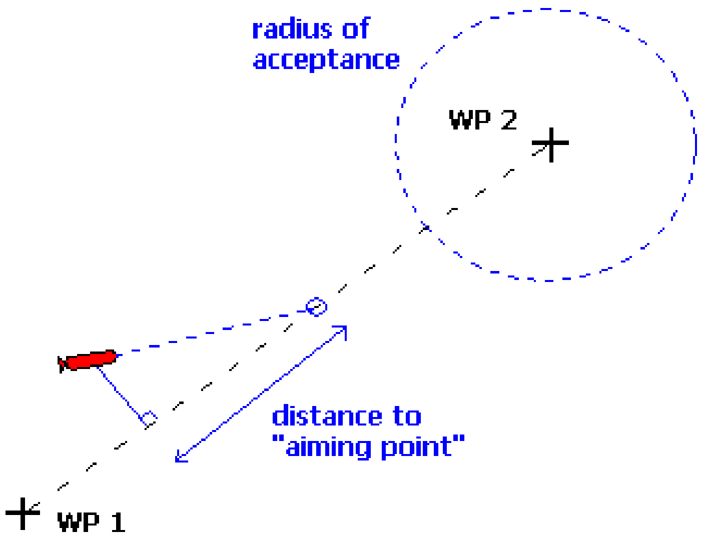


Figure 6.1: Line Of Sight Guidance

6.1 Developing the LOS Guidance System

The desired heading may be calculated in many ways and one alternative is by homogen transformation. A new coordinate frame is defined by the previous and the next WP. The vessel position is then transformed into this frame and the along track distance d_a and the cross track distance d_x may be found as shown in 6.1.1. Here θ_{wp} is the yaw angle from previous WP to the next WP and (x^n_{wp1}, y^n_{wp1}) is the position of the previous WP in NED coordinates.

$$\begin{bmatrix} d_a \\ d_x \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta_{wp}) & -sin(\theta_{wp}) & x^n_{wp1} \\ sin(\theta_{wp}) & cos(\theta_{wp}) & y^n_{wp1} \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x^n_{vessel} \\ y^n_{vessel} \\ 1 \end{bmatrix} = (A^n_{wp})^{-1} \begin{bmatrix} x^n_{vessel} \\ y^n_{vessel} \\ 1 \end{bmatrix} \tag{6.1.1}$$

The aiming point (x^n_a, y^n_a) is calculated by adding *m* meters to the along track distance and transform this back to the NED coordinate system as shown in equation 6.1.2.

$$\begin{bmatrix} x^n_a \\ y^n_a \\ 1 \end{bmatrix} = A^n_{wp} \begin{bmatrix} d_a + m \\ 0 \\ 1 \end{bmatrix} \tag{6.1.2}$$

The desired heading is then simply found by equation 6.1.3, where atan2(y,x) is the four quadrant inverse tangent function.

$$\psi_d = atan2(y^n_a - y^n_{vessel}, x^n_a - x^n_{vessel}) \tag{6.1.3}$$

In addition to this, the guidance system must keep track of how many turns around the compass the vessel has made (360° crossing). It also needs to take the vessel heading into consideration as to make sure it is making the shortest turn towards the aiming point. For details, see the matlab source code (*LOS.m*) which is included on the CD.

6.2 Simulations and Comments

The LOS guidance was tested by designing a horizontal WP route an letting the complete model with LOS guidance follow this path. The result of the test run is shown in figure 6.2. The 'distance-to-aiming-point' was set to 15 meters and the 'radius-of-acceptance' to 10 meters during the run. As shown in the figure, the LOS guidance tracks the WP path

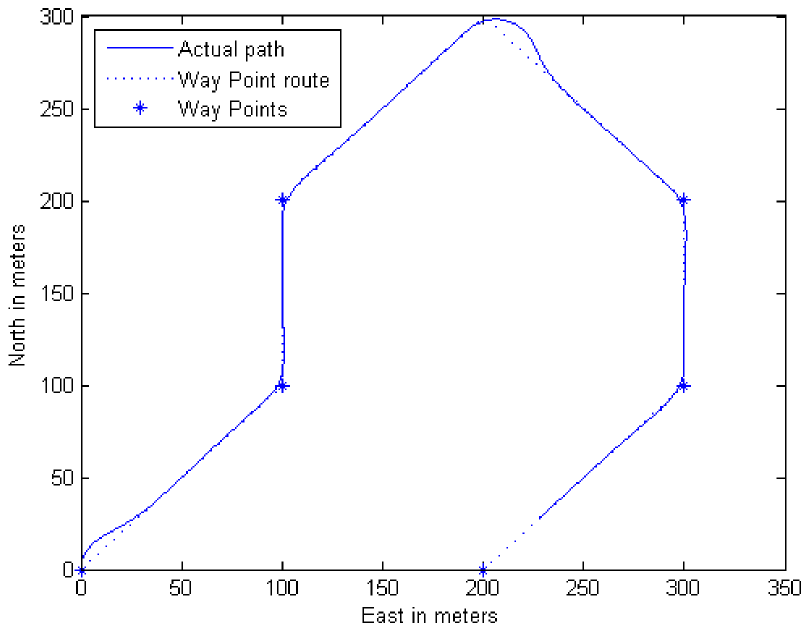


Figure 6.2: Testing the LOS guidance

relatively well. The 'distance to aiming point' and 'radius of acceptance' could be tuned to obtained different tracking. However, the values used in this test run (15m and 10m respectfully) proved to provide sufficient performance and was used throughout the work

Chapter 7

Bottom follower

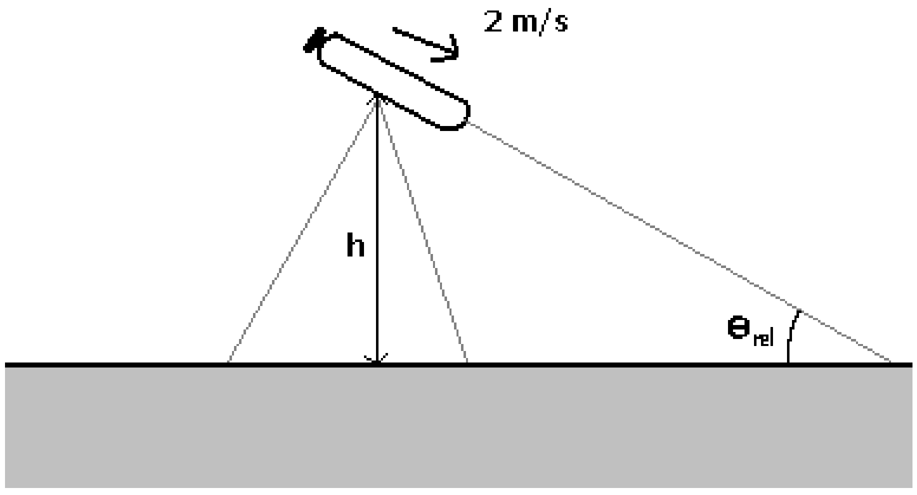


Figure 7.1: The bottom follower problem

The bottom follower implemented on HUGIN seeks to maintain a constant altitude above the ocean floor by using feedback from the vertical altimeter and feedforward from the forward altimeter as shown in figures 2.6 and 2.7. In order to produce realistic simulation results, a similar functionality needed to be implemented in the simulator model. The bottom follow control problem is illustrated in figure 7.1 where θ_{rel} is the relative angle between the current pitch angle and the bottom gradient and the distance h is the controll variable. Note that the distance h is not the same as the measured distance from the vertical altimeter.

7.1 Developing the Bottom Follower

Currently on HUGIN, the angle θ_{rel} is calculated by trigonometric relationships between the vertical- and the forward altimeter. As the FLS is introduced, the center beam may also be used for angle calculations as shown in figure 7.1. As shown in figure 7.2, this calculation

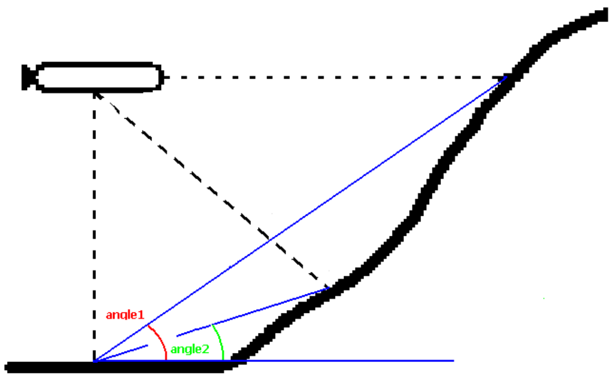


Figure 7.2: Calculating the relaive angle θ_{rel}

would lead to two potentially different angles. By choosing the steepest of the two gradients, the bottom follower would be provided with an "earlier warning" on upcoming gradients making it able to handle steeper gradient transitions. The relative distance h may then easily be calculated by equation 7.1.1, where h_v is the vertical altimeter range. This may be considered as an output linearization of the measurement.

$$h = h_v cos(\theta) \tag{7.1.1}$$

The fundamental choice in designing the bottom follower is whether to control the rudders directly or to design an outer control loop feeding reference pitch angles to the pitch-autopilot. The last option is what is called *cascade control*. In order to avoid switching between different controllers actuating the rudders, the *cascade control* design was chosen. For this reason a model of the closed loop pitch dynamics was derived. The linearized open loop pitch dynamics was in section 5.1 derived as;

$$\dot{q} = -\frac{M_{uq}|u_0|q}{I_y - M_{\dot{q}}} - \frac{Wz_g\theta}{I_y - M_{\dot{q}}} + \frac{\tau_q}{I_y - M_{\dot{q}}} \tag{7.1.2}$$

and

$$\dot{\theta} = q \tag{7.1.3}$$

Inserting the linear PD feedback in the autopilot, yields the closed loop dynamics.

$$\dot{q} = -\frac{(M_{uq}|u_0| + K_d)q - (Wz_g + K_p)\theta + K_p\theta_{ref}}{I_y - M_{\dot{q}}} \tag{7.1.4}$$

Here K_p and K_d are the proportional and derivative gains of the pitch-autopilot and θ_{ref} is the reference pitch input. From figure 7.1 one may derive the relationship between the distance h and the relative pitch angle θ_{rel}, where u_0 is the constant surge speed (2 m/s).

$$\dot{h} = u_0 sin(\theta_{rel}) \tag{7.1.5}$$

Linearizing this equation about the relative pitch angle 0 (when the AUV is running in parallel with the bottom) yields equation 7.1.6.

$$\dot{h} = u_0\theta_{rel} \tag{7.1.6}$$

Combining equations 7.1.6 and 7.1.4 yields the state space relative altitude dynamics for the AUV:

$$\begin{bmatrix} \dot{h} \\ \dot{\theta}_{rel} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 & u_0 & 0 \\ 0 & 0 & 1 \\ 0 & -\frac{K_p}{I_y - M_{\dot{q}}} & -\frac{M_u q u + K_d}{I_y - M_{\dot{q}}} \end{bmatrix} \begin{bmatrix} h \\ \theta_{rel} \\ q \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{K_p}{I_y - m_{\dot{q}}} \end{bmatrix} \theta_{ref} \tag{7.1.7}$$

In this state space model, the effect of gravity has been neglected as this relies on absolute pitch θ which is not included in this model. However, as the gravity only appears together with the proportional gain K_p in (7.1.4) and since $W z_g << K_p$, this exclusion will have minimal effects on the feedback design. Based on this state space model, a feedback controller was designed. Proportional- and derivative control was chosen for this purpose as shown in equation 7.1.8, where h_{ref} is the constant altitude reference input, L_p the proportional gain and L_d the derivative gain.

$$\theta_{ref}^{fb} = -L_p(h - h_{ref}) - L_d \theta_{rel} \tag{7.1.8}$$

The gains were found by pole-placement (Chen (1999)), using the model (7.1.7) and the matlab function *place()*. In this pole placement design, the outer altitude loop was designed to have a time constant 8 times slower than the inner autopilot loop. This was chosen partly because of the added time-lag of the rudder rate limitations. The resulting step response is shown in figure 7.3. (This step response does not include the physical rudder limitations)

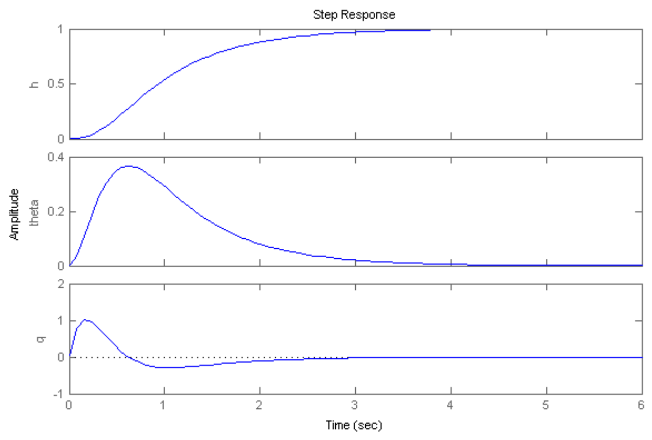


Figure 7.3: Step response of the bottom follower

Measuring the relative angle θ_{rel} relies on sonar measurements from the vertical sonar and either the FLS or the forward sonar. In those cases where the forward sonar and the FLS does not obtain measurements, θ_{rel} may not be calculated. For this reason, the θ_{rel} measurement used for feedback was instead calculated by equation 7.1.6 using approximate differentiation of the distance h, in order to reduce noise in the control loop. This insures stability even when the controller relies on vertical sonar measurements alone. Note that this is done only for the feedback part of the controller. The feedforward still uses the actual measured θ_{rel} (se figure 7.4).

As the control law operates relatively around the angle θ_{rel}, measured by the sonars, this angle must be forwarded into the control. Also, θ_{rel} is a relative measurement between the AUV pitch angle and the bottom gradient. Since the pitch controller needs an absolute pitch, the current pitch angle θ is added to the measured relative angle θ_{rel}. The resulting bottom follower control law is shown in 7.1.9 where $\theta_{desired}$ is the desired pitch passed on to the autopilot. The complete block diagram of the bottom follower is shown in figure 7.4.

$$\theta_{desired} = \underbrace{\theta_{ref}^{fb}}_{feedback} + \underbrace{\theta_{rel}}_{feedforward} + \theta \tag{7.1.9}$$

The controller may be viewed as a feedback - feedforward controller where the PD control regulates the system to the current altitude and theta is feedforwarded in order to give the controller information on upcoming terrain. Because of the time constant of the system, pure feedback would not be able to handle steep changes in bottom gradients. Integral action in the regulation loop would counteract the feedforward and is therefore not included. When

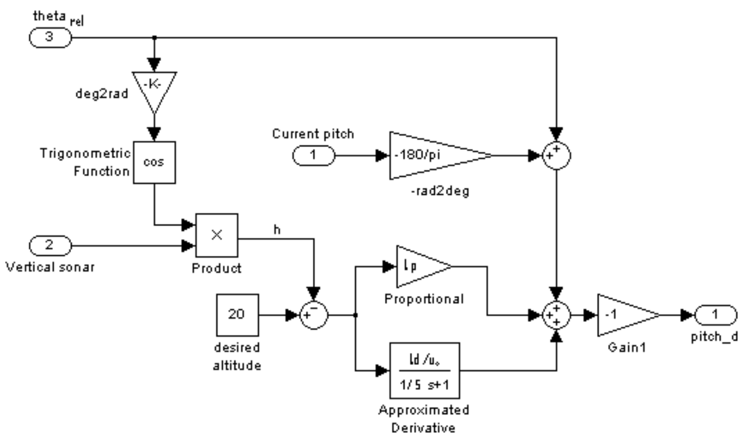


Figure 7.4: Bottom Follower block diagram

the measured angle θ is negative, the feedforward action will counteract the feedback loop by forwarding a negative pitch angle. In some cases, as shown in figure 7.5, this could lead to an altitude below the desired altitude and even collision in extreme cases. For this reason negative angles are not feedforwarded which leaves the bottom follower with pure feedback control when θ is negative. The controller will also rely on pure feedback when a measurement of θ is impossible because of limited sonar range.

7.2 Simulations and Comments

Original bottom follower

The original bottom follower, using only vertical- and forward altimeter, was tested by letting the vessel run through an example landscape with very rugged terrain. A plot of the test is shown in figure 7.6. The desired altitude was set to 20 meters during the run.

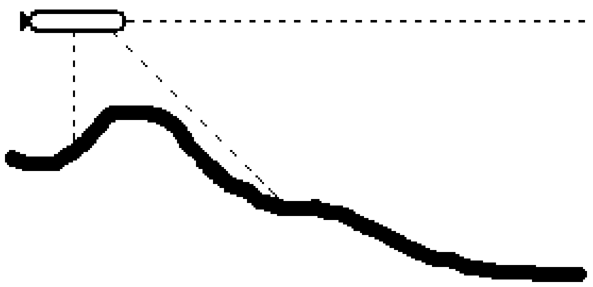


Figure 7.5: Negative measurement of angle $theta$

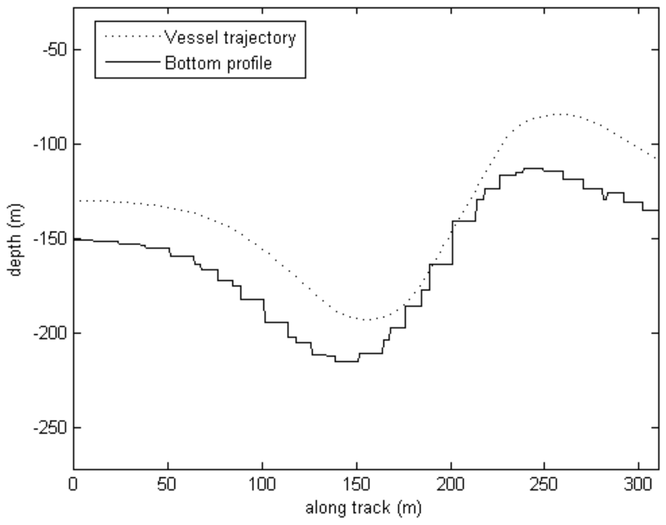


Figure 7.6: Testing the original bottom follower in very rugged terrain

This terrain is clearly too rugged for the original bottom follower to handle. Although the vessel is physically capable of avoiding collision in this case (as shown in figure 7.7), the forward altimeter does not provide information about the upcoming terrain in time for the vessel to evade. For more relaxed terrains, the original bottom follower proves excellent performance. This corresponds well with the successful track-record of the bottom follower implemented on the HUGIN AUV.

FLS aided Bottom Follower

The bottom follower was enhanced by including measurements from the FLS as described in section 7.1. This controller was then tested by letting the vessel run through the same example landscape with desired altitude set to 20 meters. The resulting plot is shown in figure 7.7

This test indicates that the inclusion of FLS measurements in the bottom follower enhances it's performance in difficult terrains. The FLS measurements give the controller a

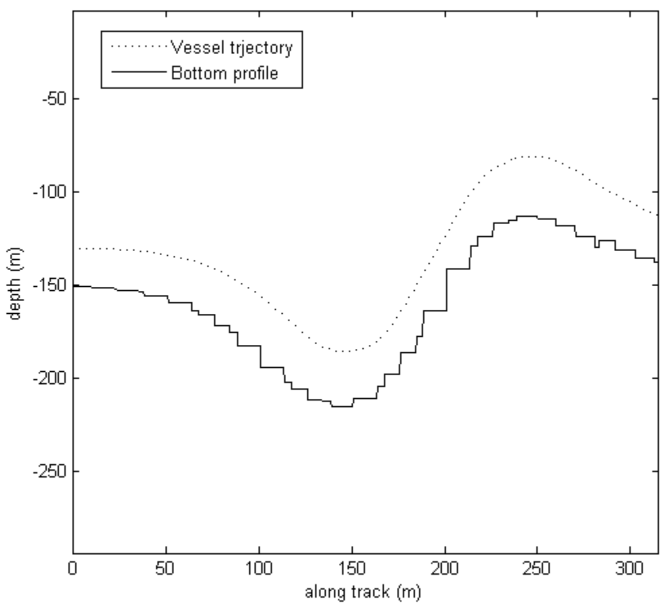


Figure 7.7: Testing the FLS aided bottom follower in very rugged terrain

’warning’ of the increasing bottom gradient, early enough for the vessel to avoid collision. Still, the distance to the bottom at the closest point is well below the desired altitude (20 m). A closer inspection of this simulation reveals that the bottom follower commanded maximum pitch rate almost instantly after the steep slope appeared on the FLS. Therefore it is concluded that the gradient transition in this terrain is simply to steep for the vessel to maintain the desired altitude using only a vertical maneuver. Further simulations revealed that the earlier warning also causes the vessel to exceed the desired altitude more often. This could have been counteracted by increasing the feedback gains but this would also increase the occurrence of negative altitude errors (and the risk of collision).In order to minimize the risk of collision, the feedback gains were left unchanged. In any case, this is open for adjustments according to user demands.

7.3 Discussion

As with the autopilot (except for the feedback linearization scheme), the bottom follower is based on linear control design. Although linear stability is guaranteed by the pole placement strategy, stability outside the point of linearization may not be guaranteed. In order to derive nonlinear stability for this controller, the complete nonlinear system must be taken into consideration including all states and internal couplings, which has proven to be quite a challenge. As the implementation of this bottom follower is only a part of developing the simulator and is not considered to be the main focus of this assignment, such an analysis is instead recommended for future work. However, the controller appears stable in all simulations providing the simulator with excellent performance in terms of maintaining the given altitude, even in very rough terrains.

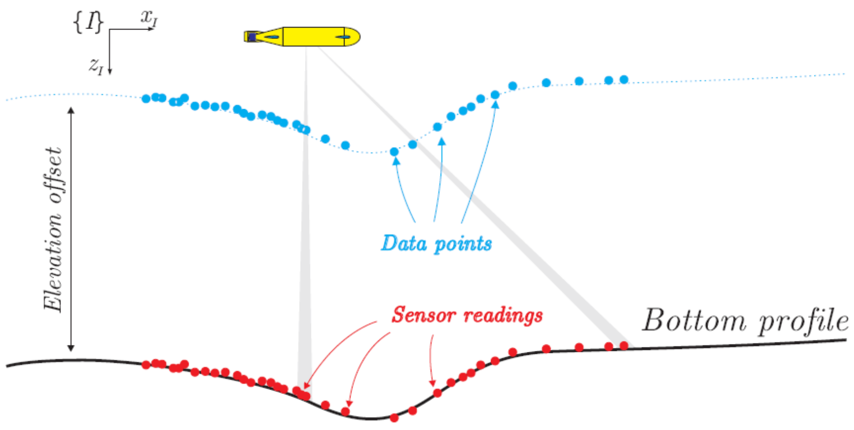


Figure 7.8: Bottom following strategy presented in Nuno et al. (2006)

In Nuno et al. (2006), an alternative strategy is presented. This design uses the forward looking sonar to create a bottom profile and then creates a reference path by adding a certain altitude to this profile (se figure 7.8). The simulations in this paper indicate excellent performance in terms of maintaining a desired altitude above the bottom. However, this differs from the controller designed in this section which seeks to maintain a constant vertical altimeter measurement. This means that the distance to the bottom in BODY coordinates should be kept constant. Using the Nuno et al. (2006) approach in high gradient terrains would lead to a correct altitude in the NED - Z direction, but the distance measured with the vertical altimeter (BODY cord) would be much less (distance = altitude$*cos$[bottom gradient]). As the goal for the bottom follower for this application is to provide a constant given distance for the payload sensors (attached to the BODY frame), the implemented controller is preferred.

Chapter 8

General Modelling and Control Discussion

In this part a complete simulation environment for the HUGIN AUV has been developed, as illustrated in figure 2.8. Although the simulator has been designed and developed around the HUGIN AUV, it may represent any AUV simply by adjusting model parameters and sensor configuration. The model including the sensor simulators and control systems has proven excellent performance in the simulations and should provide a suitable and accurate test bench for collision avoidance- development and testing. The development of this simulator has also given an indication on how the current altitude controller on the HUGIN AUV may be enhanced, using the added information from the proposed Forward Looking Sonar (FLS).

As discussed earlier in this chapter, all the implemented controllers except for the feedback linearization autopilot, are based on linear theory and design. This implies stability around the points of linearization. However, all simulations performed indicates stability within the entire operation region. Although nonlinear controllers would have been desirable for this application, this has proven not to be an easy task for this complex model with 6 degrees of freedom and multiple couplings between the states. And as AUV control is only a part of this (rather extensive) project, this is considered out of the scope of this assignment. Further nonlinear analysis (and potential control design) is instead recommended for future work, in the case of a practical implementation of any of the controllers designed in this part.

Part II

Collision Avoidance System

Collision Avoidance Overview

The objective of this part is to develop functionality that is not yet present on the HUGIN AUV. The developed Collision Avoidance System (CAS) should make the vessel capable of handle terrain which previously lead to collision. This includes very rough bathymetry with steep transitions in bottom gradients, vertical obstacles and bottom intersecting the ocean surface or surface ice.

By request, the proposed CAS should be compatible with the existing navigation system on the HUGIN AUV. Figure 8.1 illustrates how the CAS is planned to work in coherence with the existing navigation system implemented in the simulator. As long as the CAS is not active, the signals from the LOS guidance and the altitude controller are simply passed on to the autopilot.

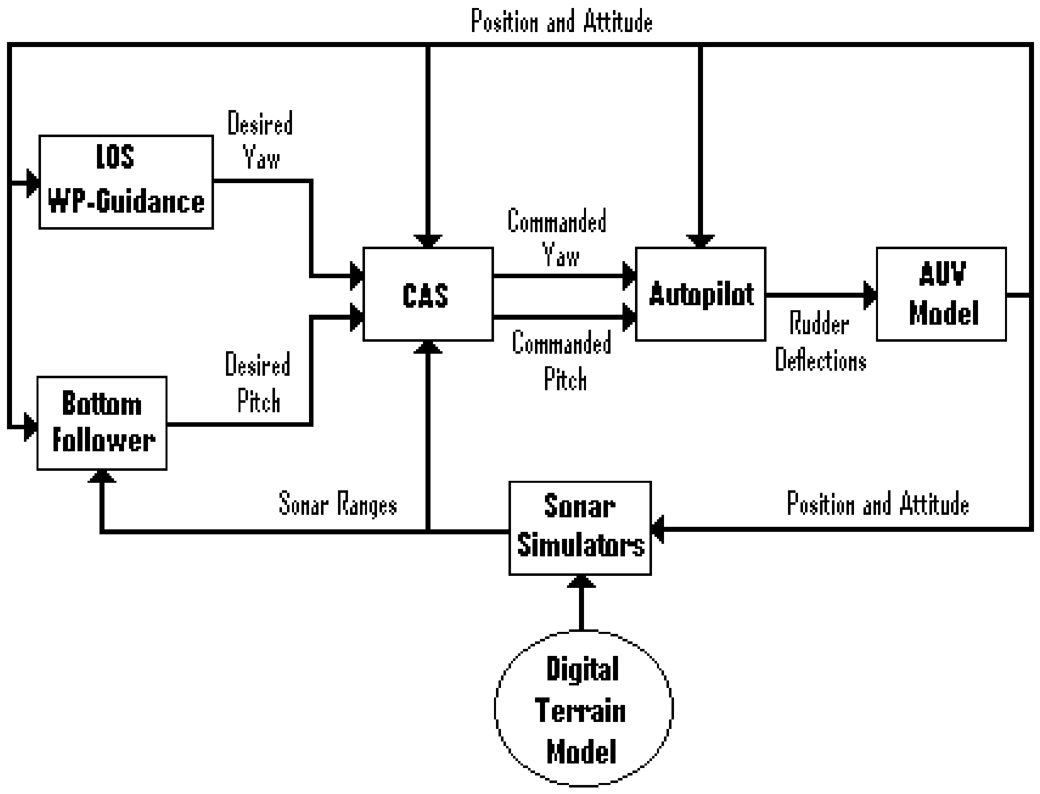


Figure 8.1: The CAS incorporated in the Simulator

Chapter 9

Collision Avoidance - a literature review

In general, AUV collision avoidance systems (CASs) are usually designed to work in parallel with the existing navigation system on board. The CAS is normally activated only when a future collision is inevitable without intervention. As a result, a collision avoidance system is to be considered as a "last-line-of-defense", and should be extremely robust in it's functionality and execution. A complete CAS system should ideally be able to handle every possible situation in an optimal way based on certain criterias. However, such a system is not likely to ever exist because of the unpredictable nature of the problem and the vast number of potential collision situations and obstacles. In addition, the information available through sensors is often very limited and corrupted with noise and errors, making it very hard to guarantee a collision free solution. Because of this, a CAS should base it's choice of actions on a combination of the available information and the probability of the different collision scenarios. In this way, the probability of a collision could be kept to a minimum. (Tan et al. 2004*a*)

9.1 Overall classification

The inclusion of a collision avoidance system in AUVs is increasing as AUV operations require an increasing degree of autonomy. Although there are many approaches to such a system, the different solutions may be characterized by their fundamental functionality. In Tan et al. (2004*a*) and Tan et al. (2004*b*) CASs are classified into three general types:

Deliberate architecture

This is a sense-plan-act approach to the CAS problem. Such a system extracts information from the sensor measurements and stores these data for later use. These data are often referred to as a World Model, as they contain information about the environment around the vessel. This allows the AUV to make reasoned decisions, predictions and plans for how to avoid upcoming obstacles. This task is often solved as an optimization problem, in order to plan a new collision free trajectory, taking the mission goals and specifications into consideration. Such an architecture is most commonly known as a motion- or path planner. The path planner scheme differs from the motion planner in the way that the resulting collision free trajectory is independent of time and may be fed into the navigation system as any preplanned path. This scheme is also potentially independent of the sensor configuration in the way that it may use all available information to create a collision free path. In many ways, this may seem like the ideal CAS solution. However, the amount of information flow from the sensor to the on board computing resources may be significant. In addition, solving such an optimizing problem is often a computational expensive task and may not be very well suited for real-time implementation. Unless the computational resources are extensive, the response may simply be to late in the case of late detection of imminence obstacles. Another disadvantage of this scheme, is that the path planning algorithms normally demands perfect and complete knowledge of the area in which the collision free path is to be planned. This is rarely the case for a surveying AUV. This architecture is employed in the *EAVE* (Blidberg et al. 1990) and the *OTTER* (Rock et al. 1995)

Reactive architecture

This architecture is a more simple, sense-react approach. It is normally based on a parallel structure where each sensor is used individually to sense on its environment and activate specific behaviors based on predefined rules. The desired global behavior is achieved by carefully adjusting each sensor and its rules to work in parallel with the others. The performance of such an architecture is often excellent when it comes to sudden obstacles and unknown environments, much because of its short reaction time and robust functionality. However the resulting collision avoidance maneuver of a pure reactive architecture is rarely optimal and it is also very sensor depended, as each sensor relies on its' own set of rules. This type of architecture is employed in the *Sea Squirt* (Bellingham et al. 1990) and the *Twin Burger* (Fujii & Ura 1996)

Hybrid architecture

This is a combination of both of the above architectures. As stated previously, pure deliberate and reactive architecture do not function adequately for all collision avoidance tasks. As an hybrid, it may combine the advantages of the two architectures and provide a more robust and a closer-to-optimal solution than any of the two alone. However, the inclusion of reactive functionality does increase the sensor dependability compared, to the pure deliberate architecture. Still, it is not surprising that the hybrid architecture is implemented in the majority recent collision avoidance systems. This is also the architecture which will be discussed in this paper. The *Garbi* (Ridao et al. 2001), the *SAUVIM* (Yuh & Choi 1999) and the *Phoenix* (Healey et al. 1995) are some of the AUVs exploiting the advantages of the hybrid architecture.

9.2 System functionality

Tan et al. (2004*a*) suggests the following functionality for a hybrid system. First of all, a target must be detected by the FLS. This information has to be interpreted and combined with the AUV navigation data such as position and attitude, to be included into the world model. From this information, a path planning technique is applied to find a new collision free trajectory based on the original trajectory and predefined criterias. Once the obstacle has been successfully avoided, the AUV should return to the originally planned trajectory. To cope with sudden or unexpected objects, a reactive submodule is included. This module normally becomes operative if it detects that the path planner is unable to avoid collision. In this case it performs a collision avoidance maneuver based on predefined logic and returns the control to the path planner.

In Tan et al. (2004*a*) and Tan et al. (2004*b*) the following decomposition of a hybrid architecture CAS is suggested:

Obstacle Detection Module
1. Forward looking sonar (FLS)
2. Sonar Processing module
3. Navigation submodule (provided by the navigation system)
4. Map builder

Obstacle Avoidance Module
1. Path Planner / Way Point (WP) generator.
2. Trajectory tracker (provided by the navigation system)
3. Reflexive submodule

The obstacle detection module is responsible for building up a world model of the known environment and obstacles using the available sensors. This information is the basis for the obstacle avoidance module to create a new collision-free trajectory. The main challenge is to interpret and extract useful information from noisy and maybe even conflicting data from the sensors. The main functionality of each submodule is given by it's name. For further details about this module and possible solutions, the reader is referred to Tan et al. (2004*a*).

The Obstacle Avoidance Module is responsible for planning and executing collision avoidance maneuvers, when the original planned trajectory is infeasible. The trajectory tracker submodule is often provided by the standard navigation system which is assumed already implemented on the AUV. But in some cases, the trajectory planning and execution may be performed by the same functionality. Details and possible solutions for the two other submodules are given in the two following subsections.

9.2.1 Path planning

In (Tan et al. 2004*b*) the general path planning problem is defined as:

> *Find a continuous sequence of configurations that leads from a start- to a goal- configuration, while respecting certain constraints.*

This problem has been covered widely by the robot community for decades and there is plentiful of literature regarding possible solutions. However, for AUV purposes the number of suitable algorithms decreases significantly as the AUV path planning problem differs somewhat from the general case. Most classic path planning approaches assume a completely known environment. Unfortunately, this is rarely the case for an AUV. In addition, an AUV operates in 3 dimensions while most of the path planning solutions are based on a 2 dimensional space. However, many of the algorithms may be extended to the 3D-case in exchange for increased computational complexity. For an AUV collision avoidance system, the path planning problem may be considered as a search in state space for a control input that can bring an AUV from an initial position to a goal position, respecting the given constraints. Following are some of the most promising path planning approaches already exploited for AUV collision avoidance purposes.

Cell decomposition

Cell-decomposition is one of the most popular path planning schemes. The basic principle is to the separate the entire search space into free- and blocked- cells. The connection between adjacent free cells are represented as a connectivity graph and this may be searched by various algorithms to find a path that connects the start cell with the goal cell. A few of the most popular ones are breadth-first search, depth-first search and shortest distance algorithms (e.g. Dijkstra). In many cases, especially for the 3D case, searching the entire search space can simply be too computationally-demanding to be implemented on an AUV. Therefore some heuristically-enhanced versions have been deviced. One of these involve using the Euclidean path (line-of-sight) as a starting point for the search. An approach that uses this principle and a combination of breadth-first and depth-first is called A* (Hart et al. 1968). In Hyland (1989) and Hyland (1990), this algorithm is implemented as a 3D path planner in a simulated AUV model. Other versions of algorithms based on cell decomposition have also been implemented for AUV path planning purposes.((Allison et al. 1989),(Aria et al. 1994),(Arinaga et al. 1996)) (Tan et al. 2004*b*)

Potential field

The potential field method uses a very interesting approach and has become very popular within AUV collision avoidance. Basically, the environment around the AUV is represented by an artificial potential field. Obstacles are represented as repulsive fields and the goal as an attractive field. The total potential field is the combination of the repulsing obstacles and the attractive goal. An example of such a field is given in figure 9.1. Eventually, the vehicle is required to just follow the local gradient of the combined field to reach the goal position, much like a ball continuously rolling in the direction of the slope. The field of artificial forces $\vec{F}(q)$ in C is produced by a differentiable function, $U : C_{free} \rightarrow R$ with:

$$\vec{F}(q) = -\vec{\Delta}U(q) \tag{9.2.1}$$

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{9.2.2}$$

Here, U_{att} is the attractive potential field of the goal position q_{goal} and U_{rep} is the repulsive

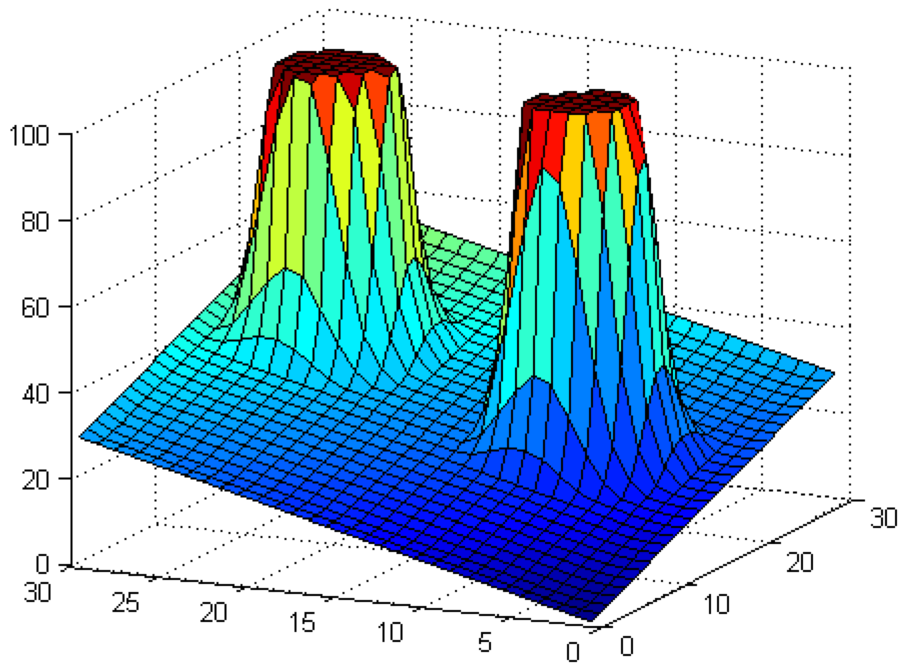


Figure 9.1: Potential field with 2 obstacles and goal position = (0,0).

field of the obstacles. The attraction field may be derived as:

$$U_{att}(q) = \frac{1}{2}\xi\rho^2_{goal}(q) \tag{9.2.3}$$

The attraction force of each position may then be stated:

$$\vec{F}_{att}(q) = -\xi(q - q_{goal}) \tag{9.2.4}$$

where ξ is the positive scaling factor, ρ is the euclidean distance, q is the current position and q_{goal} is the goal position. The repulsive field of the obstacles may be derived as:

$$U_{rep}(q) = \left\{ \begin{array}{cc} \frac{1}{2}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & if \quad \rho(q) \leq \rho_0 \\ 0 & if \quad \rho(q) > \rho_0 \end{array} \right\} \tag{9.2.5}$$

And the repulsion force may be written as:

$$\vec{F}_{rep}(q) = \left\{ \begin{array}{cc} \eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q)}\vec{\Delta}\rho(q) & if \quad \rho(q) \leq \rho_0 \\ 0 & if \quad \rho(q) > \rho_0 \end{array} \right\} \tag{9.2.6}$$

where ξ is a positive scaling factor, ρ is the distance to the obstacle and ρ_0 is the minimum distance of influence from the specific obstacle.

One major advantage of the potential field method, is its' low computational requirement which makes it ideal for real-time implementation. The big problem of this approach, is the tendency to get trapped in a local minima and not reach the goal position. In order to avoid this, each obstacle field must be convex. Intersecting obstacle fields, may be replaced with a single convex obstacle field covering all of the intersecting fields. However, this may lead to an unnecessary wide avoidance maneuver. For this reason, the potential field method is most often used as a local path planner, combined with another global path planner that will be invoked when trapped. Versions of this method has been applied to several AUVs (Yoerger et al. (2000),Warren (1990) and Lane & Trucco (2000)) for path planning purposes. (Tan et al. 2004*b*)

Bug algorithm

The bug algorithm is also known as the edge following algorithm and is a simple and yet remarkable path planning scheme. The main idea is to follow the planned path until an obstacle is encountered, and then follow the edge of the obstacle until the original path is reached again. (se figure 9.2). So in general the algorithm consist of two modes; following

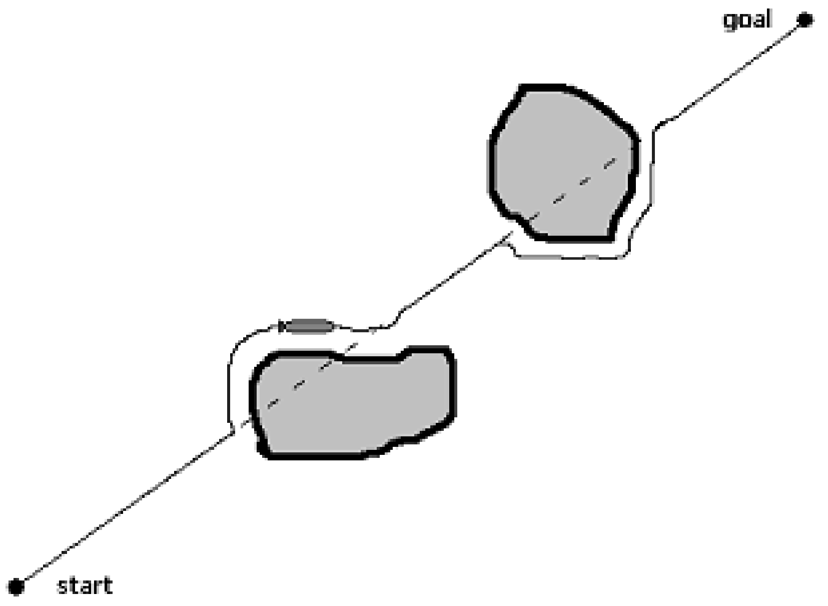


Figure 9.2: An AUV using the bug algorithm to avoid obstacles

the pre-planned path and circumnavigating obstacles. The great advantage of this algorithm, is that it does not need any a priori information about it's environment. In addition, it is guaranteed to find a solution if it exists. This makes the bug algorithm especially suited for collision avoidance in unknown environments. Although this is a very simple algorithm, it is somewhat difficult to implement in practice. Following a boundary, as illustrated in figure 9.2, requires a horizontal distance controller which in turn requires sensors for obtaining the distance to to the obstacle and upcoming distances. A horizontally mounted FLS may be an option for this purpose. In addition, performance of this algorithm in a 3D environment together with varying pitch, is somewhat hard to predict. The common implementation of this path planning scheme, bases it's decisions almost directly on sensor information and hence it could be considered a hybrid architecture. Examples of AUVs utilizing versions of this algorithm are the *Phoenix* (Healey et al. 1995) and the *Autolocus* (Cornforth & Croff 2000)

Visibility Graph

(McKendrick 1989) applied a visibility graph method in an unknown 2-D environment. This method utilizes a shortest path algorithm where the nodes are the vertices of the obstacle, represented as convex polygons. Each node in line-of-sight of each other are then connected with edges, and the shortest path algorithm is applied to find the route. However, the simulations showed that the path is highly inefficient, as such, a simple bug algorithm surpasses its performance.

Another method of placing the nodes, are by random selection and eliminating nodes that are within obstacles. This method is called *Probabilistic road-map planner (PRM)* (Tan et al. 2004*b*) and the route is found by making a visibility graph of the remaining nodes and use a shortest-path algorithm. The random placement of the nodes may be modified in order to reduce the search space. (Fox et al. 2000) In any case, the partly-random selection of nodes usually compromises the solution optimality for enhanced robustness. The PRM is also known for its long running times and difficulty in finding a path in environments with large amounts of obstacles. (Tan et al. 2004*b*)

9.2.2 Reflexive avoidance

The reflexive avoidance module is normally implemented to provide the AUV with a fail-safe mechanism in the case of path planner failure. The failure may either be a system malfunction or simply that the path planner fails to find a solution in time to avoid the obstacle. In this case, the reflexive avoidance module must take over control, and safely guide the AUV into a safe area. Technically, the reflexive avoidance module bases its actions directly on the sensor(s) and hence, belongs to the reactive architecture design. For this reason the reflexive avoidance module must be rapid in its decisions, robust and provide a fail-safe solution. (Tan et al. 2004*b*)

There are many different implementations of such a module (i.e.Demuth & Springsteen (1990),Zapata & Lepinay (1996)), but in general, they all consist of some sort of logic based decision-taking. The main difference between the implemented versions are how the 'rules' are made and how the collision avoidance maneuver is performed.

Chapter 10

Developing the Collision Avoidance System

In this section a Collision Avoidance System (CAS) has been developed and designed to be compatible with the existing navigation system implemented in part I. The CAS should be able to handle all obstacles mentioned in the assignment text. This includes rough bathymetry, vertical obstacles and bottom intersecting ocean surface. The system should also be able to handle under-ice operations. This capability puts special requirements to the developed system, especially for the fail-safe mechanisms. For operations operating in open waters, a typical fail-safe behavior would be to bring the vessel to the surface. Such situations would include hardware or software failure, a collision or simply a mission abortion. For under-ice operations, bringing the vessel to the surface is simply not an option. This environment also puts special requirements to the CAS, as the existing system is designed to pass over all encountered 'obstacles' detected by the sonars. The CAS must therefore monitor the exiting navigation system and intervene if necessary to avoid a collision with ice or other objects.

10.1 Deriving overall objective and strategy

In order to develop an appropriate collision avoidance strategy, the overall objectives should be stated. To derive this, the normal operating pattern of HUGIN needed to be taken into concern. The primary task of this vessel is to gather information about the environment using the payload sensors, e.g. Multi Beam Echo sounder (MBE) or Synthetic Aperture Sonar (SAS). The user specifies where this information should be gathered by programming a way-point route into the navigation controller. Hence, a primary goal for the collision avoidance system should be to minimize the cross track error (xte) from the way-point route during collision avoidance maneuvers. In addition, the user specifies a desired altitude above the bottom for each leg. Various altitudes give different properties for the gathered survey information. In order to act in accordance with the user's specifications, the error in altitude should also be minimized during collision avoidance maneuvers. A natural priority of these goals would be to make measurements in the desired position (minimize cross track error) and then try to make the measurements as specified (minimize error in altitude). As a result of this, the overall goals for the collision avoidance system were derived in the prioritized sequence as given in table 10.1.

1. **Avoid collision and the violation of a minimum critical distance to any object. (defined by the user)**
2. **Minimize horizontal cross track error from the planned Way Point route.**
3. **Minimize errors from the desired altitude on the current leg. Prioritize negative errors.**

Table 10.1: Overall goals for the Collision Avoidance System

The minimum critical distance was introduced to give the system added robustness. Setting this to zero will result in a pure collision avoidance behavior and may cause the vessel to pass obstacles very close and make the system very sensitive to noise and errors. As a consequence of the overall system goals, the collision avoidance strategy shown in table 10.2 was derived.

1. Attempt to pass over any obstacles encountered up to a specified distance to the surface or to the surface ice.
2. If this is not possible, circumvent the obstacle keeping a minimum distance to the way-point route.

Table 10.2: Overall strategy for the Collision Avoidance System

In most situations and conditions the current system (LOS guidance and bottom follower) is able to follow the pre-planned path without violating the critical distance. In fact, with the addition of the center FLS measurement in the bottom follower controller (chapter 7) the following assumption is made:
The altitude controller is able to maintain an altitude above the critical distance as long as it is possible with respect to physical constraints (pitch and pitch rate) and maximum sonar ranges.
An analytical proof of this statement is not straight forward to derive since it does not concern stability. However, the statement has proven to be very likely based on abundant simulations. The remaining cases may be listed as follows as shown in table 10.3. Clearly, in these cases the CAS must be activated in order to avoid collision.

10.2 Reflexive avoidance

As described in the literature review (chapter 9), the normal functionality of the reflexive avoidance module is to provide the CAS with a fail-safe solution in case of motion planner failure based on robust logic based decisions. As a consequence of the defined overall goals

1. The pitch rate constrains makes it impossible to maintain an altitude above the critical distance
2. The maximum pitch constrains makes it impossible to maintain an altitude above the critical distance
3. Critical distance is already violated
4. The altitude between the ocean floor and the surface (or surface ice) is to small to maintain desired distances to both objects.

Table 10.3: Fail-cases for the altitude controller

, the motion planner in this system should not be activated unless it is absolutely impossible to pass over the obstacle. This would be the case when there is less water between the surface or surface ice and the obstacle than a specified minimum, which corresponds to case 4 in table 10.3. In all other cases the system should bring the vessel over the obstacle. For this reason, the reflexive avoidance functionality plays a somewhat different part in this CAS then what is suggested in Tan et al. (2004*a*); In the 3 first cases of altitude controller failure, the reflexive avoidance module should be activated and bring the vessel out of the current situation and into another situation where it is potentially possible to pass over the obstacle. A proposition for such a maneuver is described in table 10.4

1. Turn $180°$.
2. Go back x meters and climb y meters.
3. Turn $180°$ again and return to planned path.
4. Return control to the navigation system.

Table 10.4: Reflexive avoidance maneuver

The direction (port/starboard) of this reflexive avoidance maneuver is decided based on side scan sonar measurements at the time of activation; The side with the largest measurement is chosen to be the direction of the maneuver. The most critical part of the reflexive avoidance module are the activation criterias. These criterias must be carefully set so that the reflexive avoidance maneuver is performed only when it is absolutely necessary. In addition they must cover all 3 first cases of altitude controller failure. This may be solved by dealing with each of the cases separately.

10.2.1 Case 1 - pitch rate constraints

This criteria may be constructed by considering the radius at maximum pitch rate and assuming the bottom gradient is constant within the area. In addition, some meters needs to be added in order to count for the time lag in obtaining the maximum pitch rate. Using the model and simulating the worst case; when the vessel is already in max negative pitch rate, this radius was found to be 30 meters. One must then derive if this radius will violate the critical distance. The problem is illustrated in figure 10.1. If the circle representing the pitch

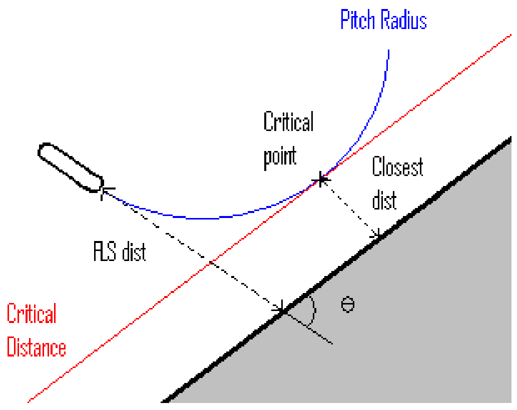


Figure 10.1: The pitch rate constraint case

radius intersects the critical distance, then reflexive avoidance should be activated. The coordinates of the closest point with reference to the vessel may be derived by equation 10.2.1, where x represents the distance along the current pitch, y represents the distance abeam the current pitch and r is the pitch radius. Again, θ is the relative bottom gradient.

$$x = rsin(\theta) \quad y = r - rcos(\theta) \tag{10.2.1}$$

The closest distance (d_c) may then be derived by equation 10.2.2 where FLS_{dist} is the distance in front of the vessel measured by the FLS. An comparison of this distance with the critical distance yields the first activation criteria of the reflexive avoidance module.

$$d_c = \left(\sqrt{(FLS_{dist} - x)^2 + y^2}\right) sin\left(\pi - \theta - atan\left[\frac{FLS_{dist} - x}{y}\right]\right) \tag{10.2.2}$$

10.2.2 Case 2 - pitch constraint

This may seem like a simple criteria by just comparing the absolute bottom gradient with the pitch constraint. However, using this as an activation criteria, the reflexive module would trigger for every bottom gradient larger than the pitch constraint even if the critical distance would not be breached. For this reason, an alternative triggering strategy is chosen. The criteria is set as follows: If anything is detected within the critical distance on the FLS while the vessel is at maximum pitch, the reflexive avoidance module is triggered. In this case, the altitude controller simply cannot avoid violating the critical distance since the pitch is already maxed out.

10.2.3 Case 3 - Critical distance already breached

If one of the measurements are already within the critical distance, it is clear that the altitude controller has failed and the reflexive module must be activated.

The complete reflexive avoidance matlab script *reflexive.m* is included on the CD.

10.3 Edge follower

In the last case of altitude controller failure, the obstacle must be circumvented. In order to do this, a motion planner strategy needs to be implemented. Most of the path-planning algorithms described in the literature review (chapter 9), require a perfectly known operation area. In fact, the edge following algorithm is the only known path planning scheme which does not require any previous information. Another property of this algorithm is that it continuously maintains a minimum distance to the desired way-point route given proper implementation and correct choice of circumventing direction (left/right). Most of the other motion planners does not minimize the cross track error, instead they are optimized to obtain a shortest possible path from the start position to the goal position. For this reason the edge follower (bug) algorithm is chosen as the motion planner strategy.

Although the edge following algorithm it selves does not require any prior knowledge of the operation area, the choice of circumventing direction does require perfect knowledge in order to guarantee optimality in the sense of minimizing cross track error. Since the area is unknown, this choice has to be made based upon sensor information. The implemented algorithm chooses the direction in which the side scan sonar is largest. This is probably the best choice one can make based on the available information.

As the yaw dynamics of the HUGIN AUV are identical to the pitch dynamics, except for the gravity term, the edge follower problem is very similar to the altitude control problem. Infact, as the gravity term was neglected in the altitude controller design, the implemented feedback design is identical. Instead of repeating the same derivation, the reader is referred to section 7.1 for details.

The resulting control law for the edge follower may then be stated as:

$$\psi_{desired} = \underbrace{-L_p(d - d_{ref}) - L_d}_{feedback} + \underbrace{\psi_{rel}}_{feedforward} + \psi \qquad (10.3.1)$$

where L_p and L_d are the proportional- and derivative gains, d is the linearized distance to the obstacle, d_{ref} is the desired obstacle distance and ψ_{rel} is the measured relative (horizontal) gradient to the obstacle. The measurements of d and ψ_{rel} however, differ somewhat from the altitude controller. During edge following the vessel should maintain the desired distance to obstacle not only straight out to the side, but also downwards. Also, in some cases the distance between the surface and the bottom is above the desired bottom-altitude but not high enough to keep both desired bottom-altitude and desired surface/ice- distance. In these cases the edge follower is activated but the 'edge' is infact below the vessel. The wide sector side scan sonars enables edge following even in these cases. The feedforward part of the controller must also be designed to maintain distances both straight out to the side and downwards. Figure 10.2 illustrates how the feedforward term is calculated in the edge follower. As seen in the figure, this calculation gives feedforward from both obstacles and

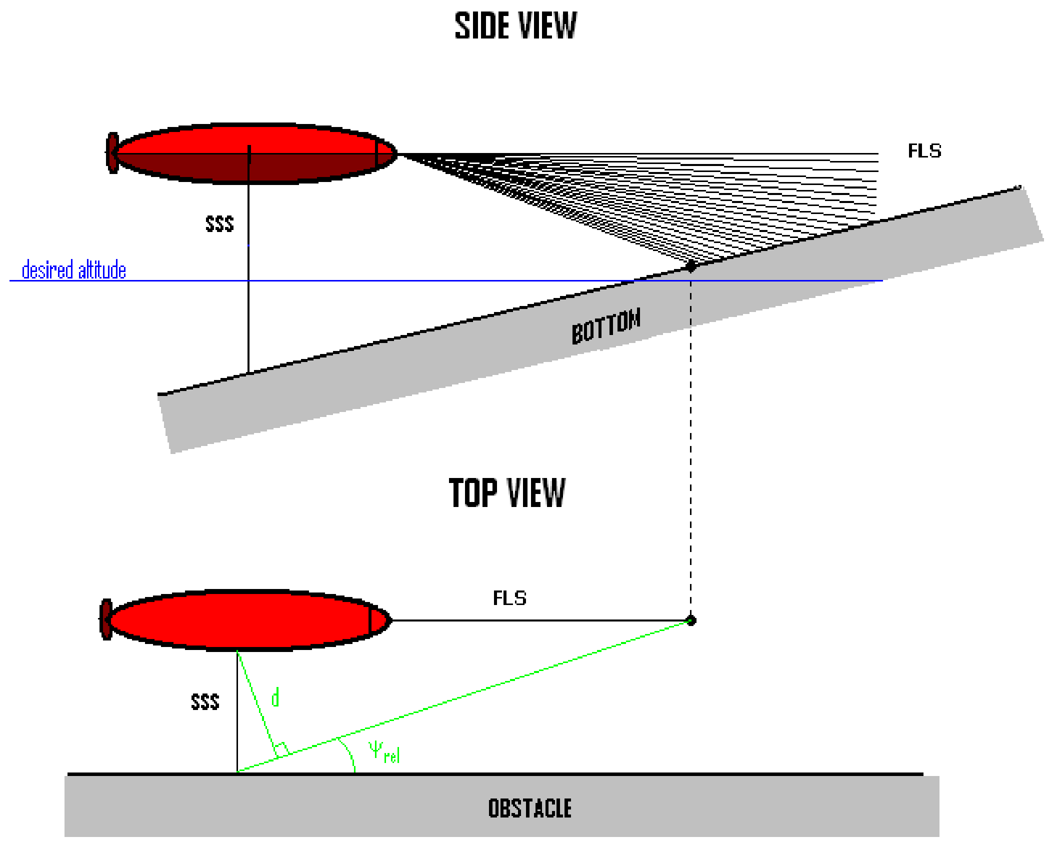


Figure 10.2: Calculating the relative horizontal angle

bottom depths less than the desired altitude. Only the lower part of the FLS is used and only bottom echoes above the desired altitude are considered in the feedforward calculation.

The edge follower activation criteria must be set so that it covers the last (4th) case of altitude controller failure (see table 10.3). For this reason, a minimum depth variable is introduced. This variable is set by the ice detection module, discussed in the next section, and represents a limit for how high in the water column the vessel is allowed to go. When the vessel approaches the minimum depth, the altitude controller is relieved by a depth controller. The depth controller regulates the vessel to the minimum depth and is active as long as the commanded pitch from the altitude controller i positive. This controller was implemented by proportional and derivative feedback from the depth obtained from the autopilot and the gains were found using pole placement. This derivation is not included here, as it is almost identical to the altitude controller developed in section 7.1. As a consequence of introducing the depth limit, the vessel must circumvent any obstacle present at this depth. The activation criteria for the edge follower may then simply be to trigger if any obstacles are detected when the depth controller is activated. The matlab code implementation of the edge follower *edgefollower.m* is included on the CD.

```
IF(Altitude control impossible because of AUV physical constraints)
⇒ Reflexive avoidance

ELSE IF(Altitude control impossible because of lacking depth)
⇒ Edge follower and minimum depth control

ELSE IF(Approaching minimum depth)
⇒ Use LOS heading and depth controller pitch

ELSE
⇒ Use LOS heading and altitude controller pitch
```

Table 10.5: The pseudo code for the entire CAS

Another issue arrives when trying to run the altitude controller simultaneously with the edge follower. In this condition, the vessel tend to 'slide down' steep gradients and the resulting trajectory may end up far away from the desired WP path. For this reason, the depth controller must stay in control during the entire edge following maneuver, even if the altitude to the bottom increases. The minimum depth variable however, may be changed by the ice-detection module if ice below the present depth is detected during the maneuver. The pseudo code for the entire CAS functionality may then be derived as stated in table 10.5.

10.4 Ice detection

The inclusion of surface ice in the collision avoidance problem brings forth a set of new challenges. Up until now, the system has been designed to pass over any encountered obstacles below the depth of the desired altitude. As surface ice may have varying depths this is no longer the case, and the limit for how high in the water column the vessel should go depends on the depth of the surface ice in that particular area. As mentioned in the previous section, the Ice Detection Module (IDM) developed in this section is responsible for setting a limit for how high in the water column the vessel is allowed to go. Setting this limit correctly will allow the current system to operate in the same way without risking collision with the ice. For this reason an ice-detection algorithm has been developed.

10.4.1 Developing the ice detection algorithm

Several approaches have been tried out in order to separate ice from bottom in the sonar measurements. Knight et al. (1981) indicates that classifying the reflective object based on the properties of the received echo, is not an easy task. The strength and nature of the received echo depends on several factors such as; angle-of-approach, water temperature, water particle content and bottom vegetation. For this reason, it is assumed that identifying ice based on the properties of the sonar echo will not provide the robustness needed for a collision avoidance system. Therefore, an alternative strategy has been developed.

The reason surface ice makes out a problem for the collision avoidance system, is that it consists of overhangs. In fact, all overhangs represent the same problem; they must be evaded by diving and not climbing. In addition to surface ice, such overhangs may consist of anything present at the ocean surface e.g. floating oil-rigs, vessels etc. Simulations indicate that overhangs of more than 70° (from horizontal) are handled well by the existing system. Such overhangs causes the reflexive avoidance module to trigger repeatedly until the vessel reaches the minimum depth. Then the edge follower module is activated and the obstacle is circumvented. For overhangs of less than 70° the system is not able to maintain the critical distance and there is a risk of collision. As a consequence, all overhangs of less than 70° are classified as ice and should be evaded by diving. The probability of encountering such overhangs on the ocean floor is considered minimal. In addition, the developed algorithm includes additional features preventing bottom overhangs to be classified as ice. In the same way, all gradients (not overhangs) of less then 70° are classified as bottom.

The ice-detection algorithm uses the FLS to classify and determine the depth of ice/overhangs by measuring the absolute gradient between adjacent sonar beams. The points of interception are rotated to the NED frame and the absolute gradient is calculated. In order to determine if the gradient is an overhang or not, the algorithm also have to calculate if the extension of the line between the two interception points crosses above or underneath the vessel. This is illustrated in figure 10.3. If the gradient is classified as ice, the ice depth is

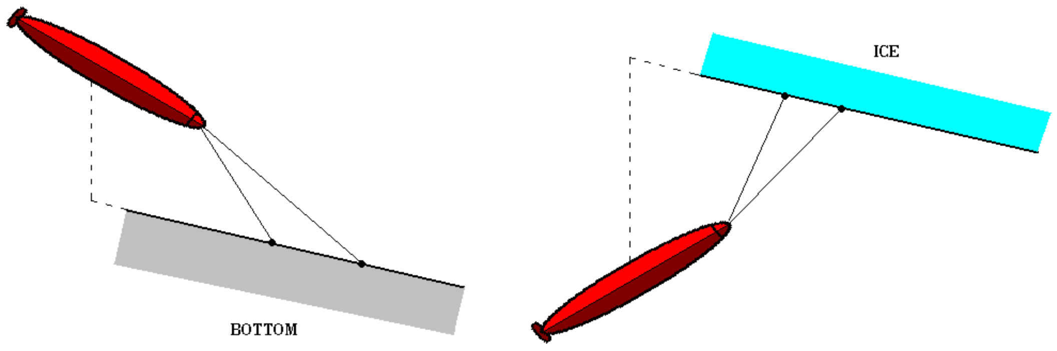


Figure 10.3: Classifying ice and bottom

set to be the depth of the lowest of the two points. As ice is most likely to be discovered by the topmost sonar beams, the algorithm starts at this end. It then iterates downwards through the sonar beams, checking for overhangs in each pair. The current ice-depth is continuously set to the deepest measurement obtained. If the algorithm finds a bottom gradient or a sonar beam that has reached it's maximum distance, the search is stopped and the lowest ice measurement so far is used as the ice depth. This is to reduce the risk of classifying ocean floor overhangs as surface ice. In the same fashion the algorithm searches through every FLS measurement and updates the current ice-depth if the latest measurement detects ice at a greater depth. The pseudo code for the ice-detection algorithm is shown in table 10.6.

The ice detection algorithm is run continuously as the vessel operates. Assuming the vessel is moving in the direction of the nose, the only way it may collide with ice is by running into it. In these situations, the ice detection algorithm should detect the ice and set the depth limit a given amount of meters below it so that collision is avoided. The only cases in

```
For all FLS beams (starting at the topmost beam):

1. IF (Current or next sonar beam have reached maximum distance)
   END SEARCH → Keep current ice depth

2. Calculate point of interception for current and next sonar beam in the
   BODY frame

3. Rotate the two interception points to the NED frame

4. Calculate absolute gradient between the two points

5. IF (Gradient < 70°)
   (a) IF (Extension line crosses above the vessel)
       Update ice depth if lowest point is below current ice depth

   (b) IF (Extension line crosses below the vessel)
       END SEARCH
```

Table 10.6: Reflexive avoidance maneuver

which the ice depth is used, is under edge following maneuvers where the depth is regulated to the depth limit. After the edge following maneuver is finished, the bottom follower is reactivated and the current ice depth is set to zero. The ice detection algorithm should detect any new occurrences of ice and set the ice depth accordingly. The implemented matlab-code ice detection algorithm *iceDet.m* is included on the CD. The simulink diagram of the complete simulator system with CAS is shown in appendix B.3 and the file *simulator.mdl* is included on the CD. For details on how to run simulations using this file, se appendix A.

10.4.2 Ice detection simulations and discussion

The ice detection algorithm was run trough a series of simulated terrains with different ice depths. In all simulations the algorithm was able to detect and determine the lowest point of the ice within 0.1 meters. The simulations in the next sections will show how the ice detection module works in cooperation with the rest of the CAS. The main concern with this algorithm is the risk of classifying bottom as ice, and setting the ice-depth accordingly. In the simulated environment this has not been a problem, but it is hard to predict how this would appear in a real environment. Sill it is believed that this method of classifying ice should provide a worthy basis for ice-detection in a real environment. A possible extension to this algorithm is to add a criteria of how long an overhang section must be in order to be classified as ice. This should prevent any small overhangs at the bottom (e.g. due to vegetation) to be wrongfully detected as ice. In any case, physical trials and adjustments should be performed prior to a full CAS implementation.

Chapter 11

Collision Avoidance Simulations

In this chapter the AUV simulator with the complete Collision Avoidance System (as illustrated in figure 8.1) was run through a series of simulated landscapes in order to test the robustness of the system. This includes all of the obstacles mentioned in the assignment text. All the simulations are included on the CD and it is STRONGLY recommended to view these plots using a computer, as they are 3 dimensional and not very well suited to be presented on paper. Still 2D representations of each simulation is included in the report, together with a plot of the along track- depth and bottom profile for some of the runs.

All of the simulated terrains are unusually rugged compared to the normal operating areas of the HUGIN AUV and for most other AUVs. Still, it is under such conditions the CAS plays a part in obtaining the mission goal. For less rugged terrains, the original navigation system does not need CAS functionality to operate safely, hence simulating such conditions is not very interesting when evaluating the CAS. All simulations in this chapter have been performed with the parameters set according to table 11.1. Se chapter 10 for details about the different parameter significance.

Desired bottom altitude	20m
Critical distance	10m
Minimum distance to ice or surface	15m
Reflexive avoidance backtracking distance	30m
Reflexive avoidance altitude increase	30m
Desired edge following distance	20m

Table 11.1: CAS simulations parameters

11.1 Rugged terrain

The first test run was performed on a landscape with rugged terrain. This included bottom gradients exceeding 60° and steep gradients transitions. First the model was run through this landscape without the collision avoidance module (With the FLS aided bottom follower). The result is shown in figures 11.1 and 11.2 and can be found on the CD (*RuggedTerrain1.fig*). Then the complete system with collision avoidance was run through the same landscape and WP-trajectory with the results shown in figures 11.3 and 11.4. This plot can be found on the CD under *RuggedTerrain2.fig*.

The simulated test run without the CAS reveals that this terrain is clearly to rugged for the original system to handle. The rapid gradient transitions combined with the steep gradients makes it impossible (because of the physical vessel constraints) to maintain desired altitude and to avoid collision based on vertical maneuvers alone. This causes the vessel to touch the bottom at about 200m along the trajectory. For such rugged terrains, a Collision Avoidance System is clearly necessary in order to fulfill the mission goals and to avoid collision.

In the simulation test with CAS the reflexive avoidance is activated just before the same section, avoiding the collision and the violation of the critical distance. Of course, such a maneuver would cause the vessel to spend more time (and electricity) because of the additional traveled distance. Even so, the overall goal is to cover the entire WP path as close to the desired altitude as possible, without violating the critical distance. Figure 11.4 shows that the altitude is very high during the reflexive avoidance maneuver but this is not a part of the WP path and the altitude in this section is therefore considered irrelevant. The non-symmetric nature of the reflexive avoidance maneuver is caused by varying pitch which affects the horizontal turning radius. The maneuver is started with maximum pitch until the desired altitude increase is reached and then continued at that altitude with zero pitch. The minimum horizontal turning radius at maximum pitch is about half of the radius when the pitch is zero.

Because of the pitch constraint, the only way to increase the 'climb rate' beyond the maximum pitch is to perform an extra loop. Alternative solutions without this loop would have to maintain higher altitudes in some sections of the WP path, leading to reduction of measurement quality in these sections.

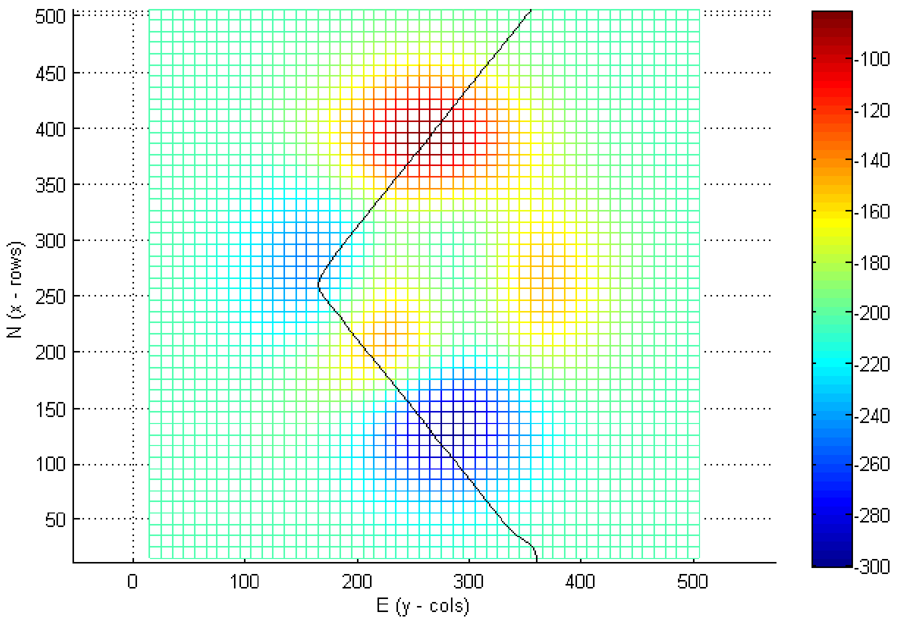


Figure 11.1: Running the system without CAS through rugged terrain

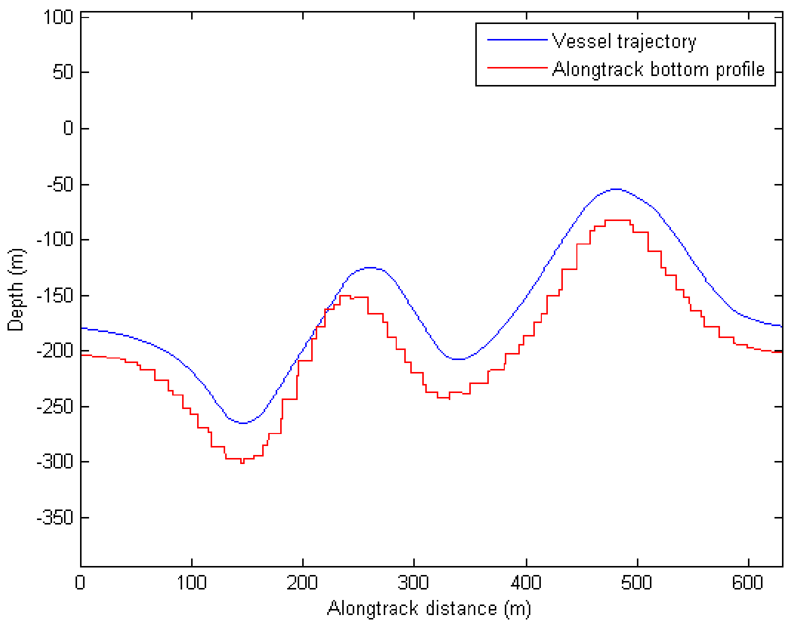


Figure 11.2: Bottomplot of the system without CAS through rugged terrain

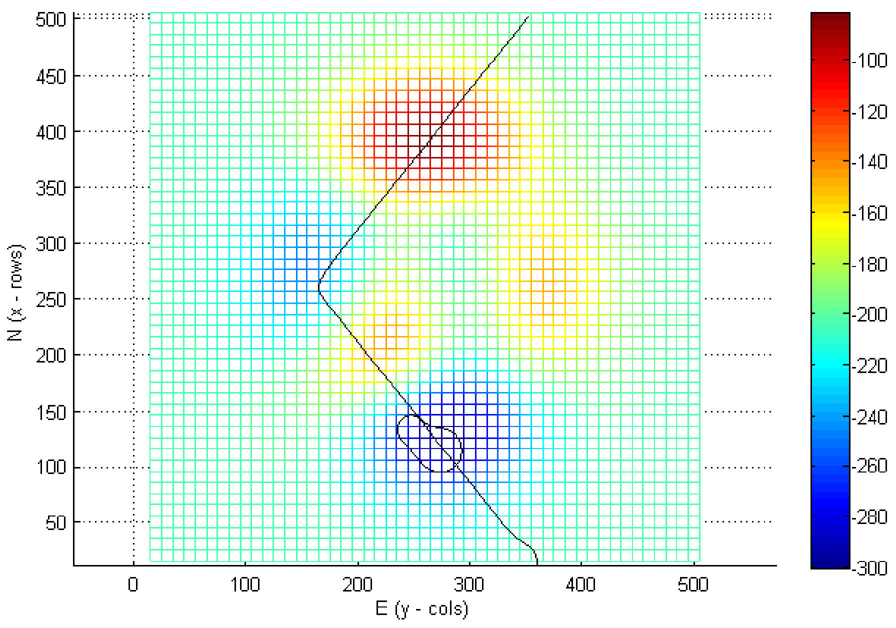


Figure 11.3: Running the system with CAS through rugged terrain

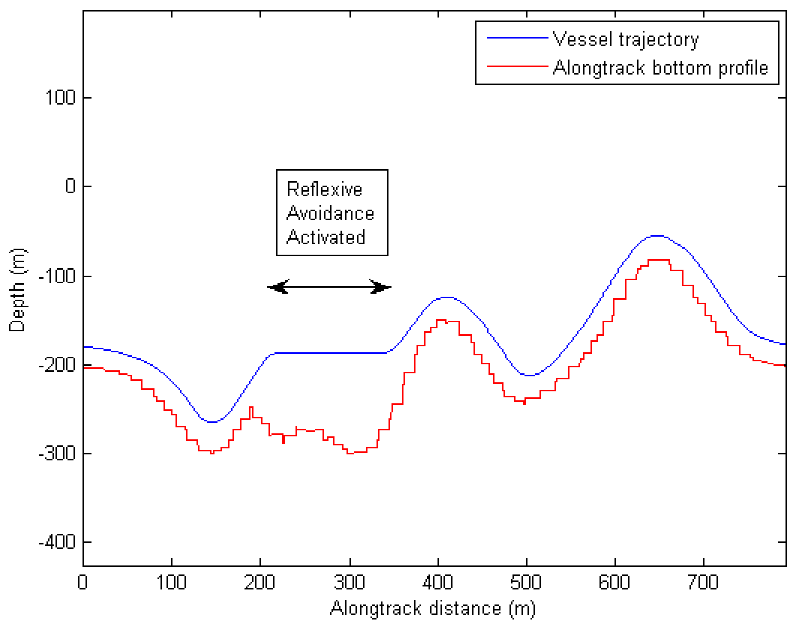


Figure 11.4: Bottomplot of the system with CAS through rugged terrain

11.2 Rugged terrain with islet

This test run was performed on the exact same landscape, only elevated so that the highest peak was above the surface forming an islet in the middle of the WP path. Only the system with the collision avoidance was run through this terrain, as the system without CAS would clearly not handle this terrain. The results are shown in figures 11.5 and 11.6. In figure 11.6, RA denotes that Reflexive Avoidance is activated and EF denotes that Edge Follower is activated. This notation is also used for the remaining simulations. The same test run is shown in the file *RuggedIslet.fig* on the CD.

As this terrain is identical to the last section, only elevated in order to make an islet, the reflexive avoidance was triggered in the exact same position. When the vessel approached the islet, the depth controller assumed pitch command as the depth approached the minimum allowed depth (15 m). When the islet was detected by the FLS, the edge follower was activated. First, the direction of the avoidance maneuver was decided based on Side Scan Sonar measurements. As the starboard side had a somewhat larger value than the port, the edge follower decided to circumvent the islet on the right side. The simulation shows that the islet is circumvented smoothly and control was passed on to the LOS guidance and altitude controller, once the vessel was back on the WP path.

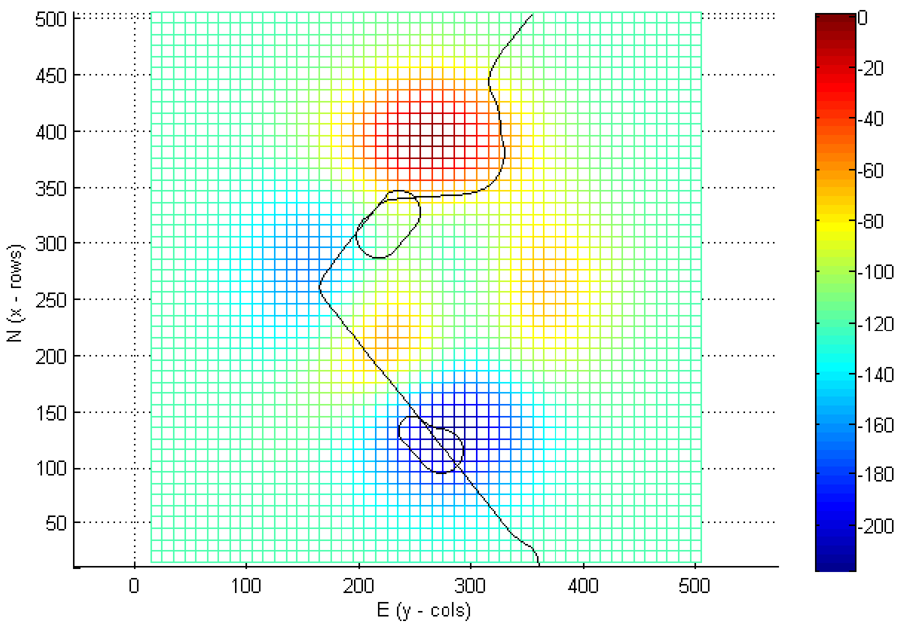


Figure 11.5: Running the system with CAS through rugged terrain with islet

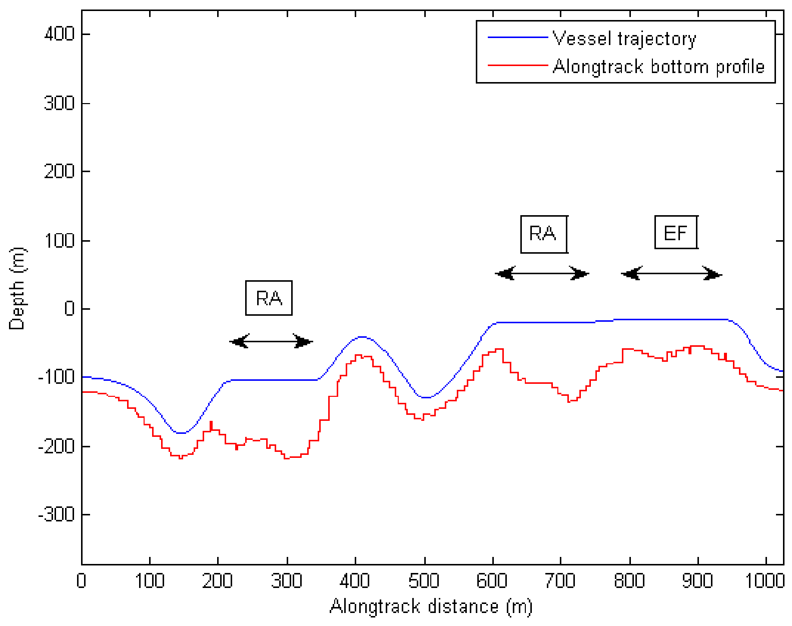


Figure 11.6: Bottomplot of the system with CAS through rugged terrain with islet

11.3 Extremely rugged terrain

The model was then run through an extremely rugged terrain with the highest peak forming a bank, to shallow for vertical avoidance. This includes bottom gradients exceeding 75° and extreme gradient transitions. The result of the model with CAS running through this terrain is shown in figures 11.7 and 11.8. The same run is displayed on the file *ExtremeTerrain.fig* on the CD.

The terrain simulated here is truly extreme and not very suited for AUV operations. Still, the simulation shows that the CAS makes the vessel capable of covering the entire feasible section of the WP path without violating the critical distance. The simulation also shows that the edge follower was able to circumvent the shallow bank smoothly, even though the 'edge' was infact below the vessel and not straight out to the side. The wide sector side scan sonars enables this capability.

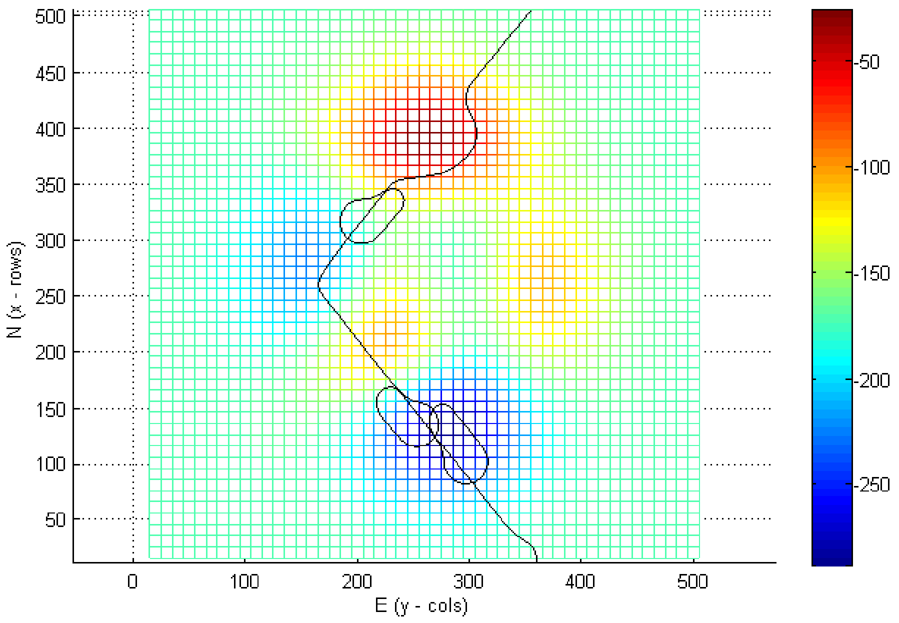


Figure 11.7: Running the system with CAS through extremely rugged terrain

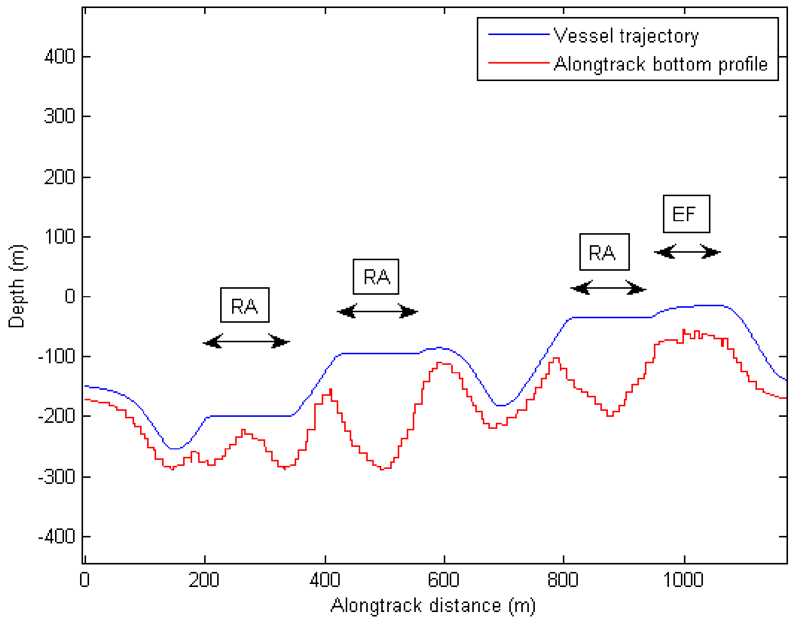


Figure 11.8: Bottomplot of the system with CAS through extremely rugged terrain

11.4 Vertical Obstacle

In accordance with the assignment text, the model was run through a terrain with a vertical obstacle. Figures 11.9 and 11.10 displays the resulting trajectory of the vessel encountering a 200m high vertical obstacle. The file *VerticalObstacle.fig* on the CD displays the same run.

When the vessel encountered the vertical obstacle, the reflexive avoidance was activated and re-activated until the minimum depth was obtained. When the FLS detected that the obstacle was still present, the edge follower was activated. In this case, the Side Scan Sonar measurements were equal and a starboard maneuver was chosen by default. As the simulation shows, the edge follower experienced some problems with the square corners of the obstacle. The feedback gain of the edge follower is very relaxed in order to avoid suppression of the feedforward. For this reason the distance controller needed some time to stabilize on the desired distance after each corner. Additional tuning of the edge follower control algorithm would probably have increased it's performance in this case. However, the system was still capable of safely circumventing the obstacle without violating the critical distance, which is the primary goal of the CAS.

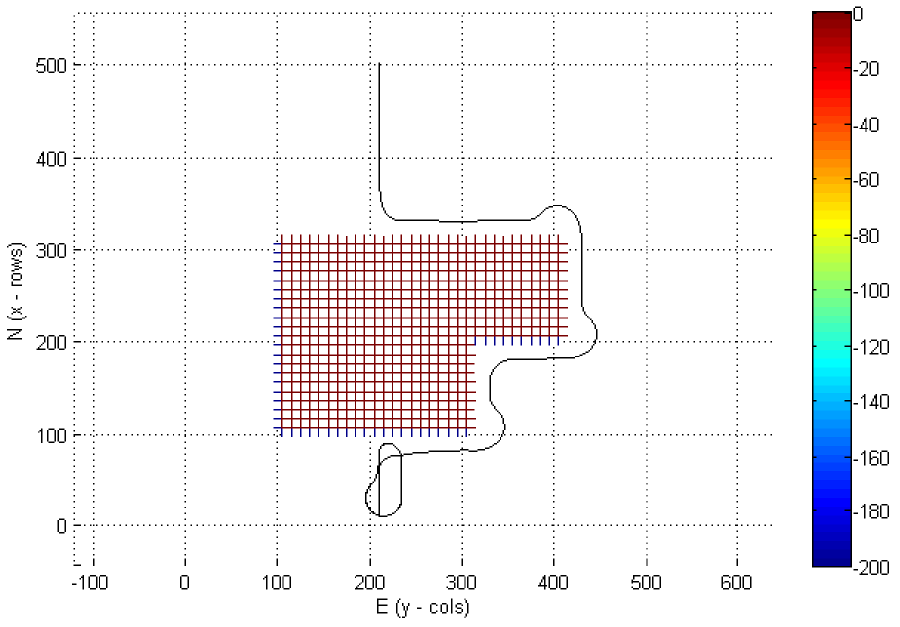


Figure 11.9: The system with CAS encountering a vertical obstacle

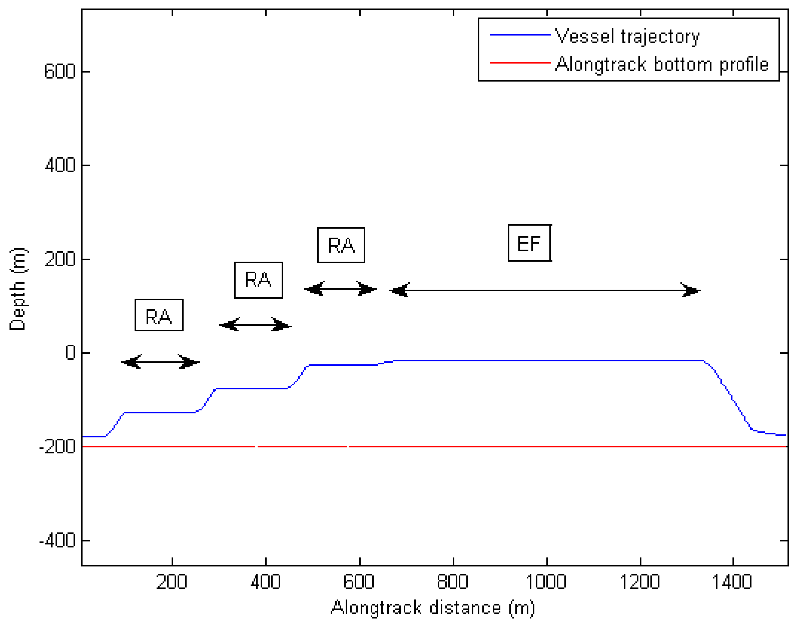


Figure 11.10: Bottomplot of the system with CAS encountering a vertical obstacle

11.5 Terrain with level surface ice

In order to test the ice detection capability of the CAS, the system was run trough a terrain covered with level surface ice at 30 meters depth. In this simulation, the ice detection module had to detect and measure the depth of the ice and set the minimum allowed depth accordingly (15m below). The result of this simulation is shown in figures 11.11 and 11.12. The same simulation is included on the CD under *LevelIce.fig*.

Although the bottom follower is able to run through this bottom-terrain without CAS interruption, the inclusion of the surface ice demands a different trajectory in order to fulfill the mission goals. As the vessel approached the the first peak, a reflexive avoidance maneuver was triggered due to the small gap between the ice and the bottom profile. Infact, this gap is to small to maintain desired distances to both bottom and ice. For this reason the edge follower was triggered and the gap was circumvented. The depth controller assumed command of the pitch during this maneuver and regulated the depth to 15m below the measured ice depth. Printouts from the ice detection algorithm showed that the module was able to determine correct ice depth within 0.1 meters. After the first edge following maneuver was finished, the ice depth was again set to zero. As the vessel approached the second (highest) peak, the ice detection algorithm once again found the correct ice depth (within 0.1 meters), and circumvented the peak at given distances to both the ice and the obstacle. This simulation indicates that the ice detection algorithm works as specified and enables the system to handle operating areas covered with surface ice.

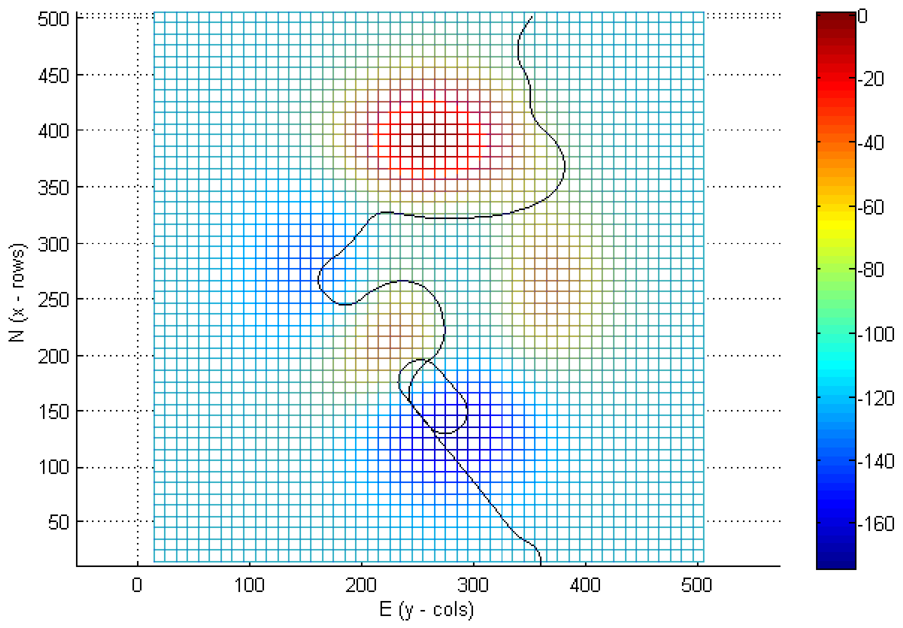


Figure 11.11: Simulation with level surface ice (viewed from above

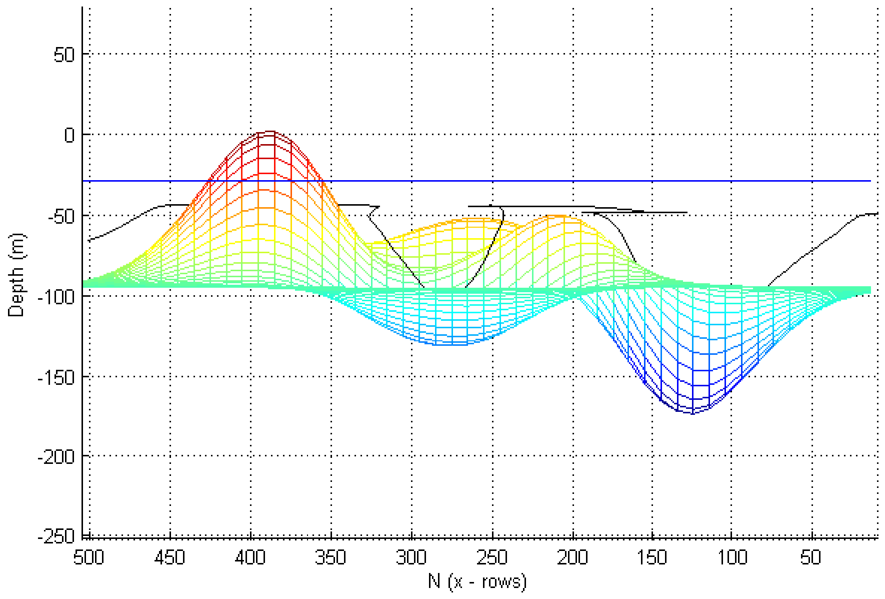


Figure 11.12: Simulation with level surface ice (lateral view)

11.6 Terrain with varying ice-depth

In order to test the ice detection module under varying ice-depths, the system was run through an identical bottom profile with undulating surface-ice. The result is shown in figures 11.13 and 11.14. The same simulation may be found on the CD under *VaryingIce.fig*

As shown in the figures, the system detected and determined the ice depths in both cases were a horizontal maneuver was necessary. The edge follower then circumvented the obstacles at the given depth below the lowest measured ice in that particular area, in order to avoid collision with either ice or ocean floor. This simulation indicates that the ice detection algorithm is able to measure the ice, even under varying ice-depths, by calculating the depth of the lowest point of the ice. The simulation also shows that the ice detection module works well in cooperation with the rest of the CAS.

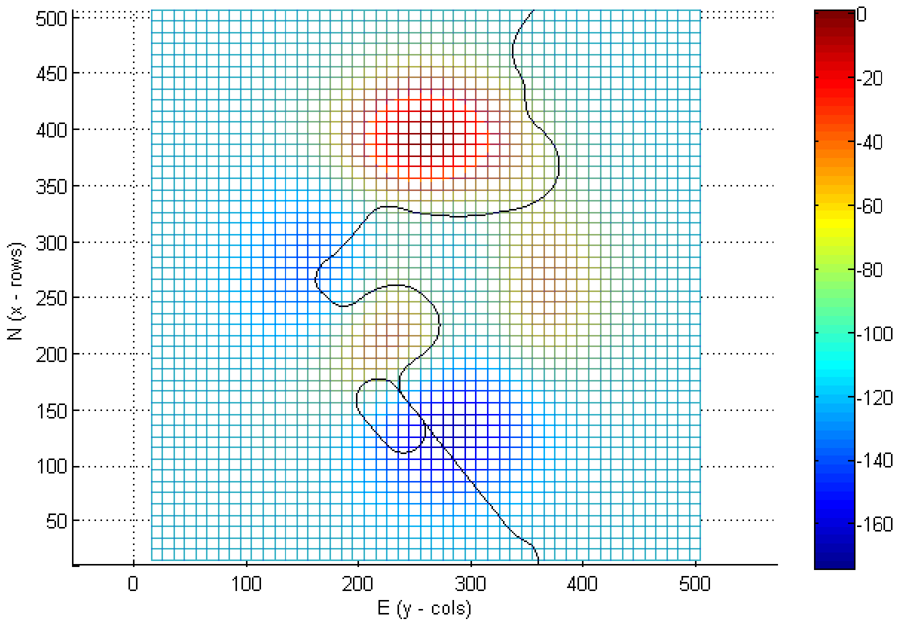


Figure 11.13: Simulation with varying ice-depth (viewed from above)

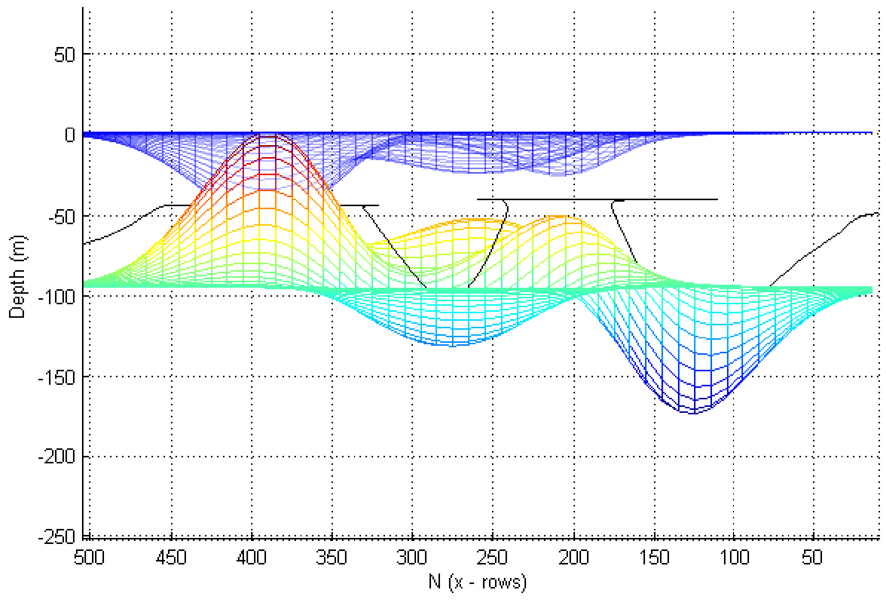


Figure 11.14: Simulation with varying ice-depth (lateral view)

Chapter 12

General discussion and concluding remarks

In all the simulations the collision avoidance system performs well and acts in accordance with the overall goals and user requirements. However, due to the complexity of the collision avoidance problem, it is very difficult to design a system capable of handling every thinkable situation. By the same reason, there are a number of situations in which the developed system will not be able to avoid collision. An example is the AUV getting trapped within a subsurface cave. However, it is assumed that the developed system will be able to handle most situations by adding functionality for special cases. For this reason, the developed system in this project should be considered only as a concept of collision avoidance. There is still a number of issues that needs attention before the system is ready for implementation.

The collision avoidance system, including all the subsystems, forms what is called a hybrid system. As described in Branicky (1998), a hybrid system combines logical and continuous processes into one dynamical system. In this system, the logic part is performed by the CAS functionality, e.g switching between the bottom follower and the depth controller. Analytical stability for hybrid system is a subject which is still under extensive research, and is in any case not straight forward to derive. Branicky (1998) suggest a stability analysis using multiple Lyapunov functions. One of the concerns with hybrid systems is the occurrence of *limit cycles*. This is normally caused by the switching logic, where the system ends up switching continuously back and forth between two controllers. In order to avoid such behavior, the switching criterias must be carefully set (Branicky 1998). During development of the CAS, this behavior was experienced several times. The criterias have therefore been carefully adjusted to avoid this problem, and no such behavior have been experienced with the final system. Further analytical analysis is considered to be out of the scope of this project, but should be looked into prior to a practical implementation.

As stated earlier, deriving analytical stability for the entire system developed in this paper relies on many factors which is considered to out of the scope of this project. Still, all of the subsystems have been stabilized in the linear sense, and the logic which switches between the different subsystem-controllers have been carefully adjusted to avoid limit cycles. Extensive simulations with the complete system indicate that the complete system is stable. For this reason it is concluded that the systems, based on the previous statements and simulations, appears stable and is able to avoid collision in the majority of situations.

Further analysis on stability for this system, and it's ability to avoid collisions, is not straight forward to derive.

The developed simulator has proven to be a suitable testbed for collision avoidance systems. The simulator may be extended in order to produce more realistic simulations e.g. by including senor noise and ocean currents. However, such factors are considered to be an implementation issue and is therefore not covered by the conceptual design discussed in this paper. The proposed collision avoidance system shows promising results in the simulated environment and is able to fulfill the defined goals in all performed simulations. The functional simplicity of the derived system enhances it's robustness which is crucial for a CAS. Additional functionality may be added in order to make the system handle potential fail-situations which may be encountered in a real environment. In any case, the developed system provides a framework for further analysis and development for the HUGIN AUV collision avoidance problem.

Chapter 13

Suggestions for future work

As mentioned in the previous chapter, there are still a number of issues that needs attention prior to a practical implementation of the proposed collision avoidance system. This includes extensions to the simulator to include some of the unpredictable factors in a physical environment, such as; ocean currents, sensor noise, drift in position and attitude estimates etc. The derived system should also be further analyzed in terms of stability and it's ability to avoid collisions. Such analysis would provide enhanced credibility towards potential customers and users but may also uncover elements of risk in the design which should be altered.

In chapter 9, several path planning strategies were presented. Some of these solutions creates a map over the covered area and uses path planning algorithms in order to develop a new path. As the HUGIN AUVs are already equipped with sensors and functionality to create map-data from the passing environment, a natural extension to the proposed system would be to use these data for collision avoidance purposes. This could, amongst other things, assist the edge follower in choosing the optimal direction of circumventing obstacles. Another problem, which is not covered in this paper, arises when a planned waypoint turns out to be at the same position as an obstacle. In these cases, a superior functionality should be added which re-plans the route or simply removes the unreachable waypoint from the current way point route. Combining these features with the existing system could provide improved collision avoidance functionality.

Bibliography

Allison, J., Watson, D. & Cook, T. (1989), 'Intelligen waypoint transiting in complex auv environment', *IEEE Int Symposium On Unmanned, Untethered, SUbmersibel Technology* **6th**, 246–257.

Aria, H., Tanie, K. & Shiroma, N. (1994), 'Time-scaling control of an underactuated manipulator', *IEEE INt Conference on Robotics and Automation* **4**, 525–536.

Arinaga, S., Nakajima, S., Okabe, H., Ono, A. & Kanayama, Y. (1996), 'Motion planning method for an auv', *IEEE Proc Symposium Autonomous Underwater Vehicle Technology (AUV)* pp. 477–484.

Bellingham, J., Consi, T. & Beaton, R. (1990), 'Keeping layered control simple', *IEEE Proc Symposium Autonomous Underwater Vehicle Technology (AUV)* pp. 3–8.

Blidberg, D., Chappel, S., Jaibert, J., abd G. Sedor, R. T. & Eaton, P. (1990), 'The eave auv program at the marine systems engineering laboratory', *Proc of the 1st Workshop on: Mobile Robots for Subsea Environments* pp. 33–42.

Branicky, M. S. (1998), 'Multiple lyapunov functions and other analysis tools for swithched and hybrid systems', *IEEE Transactions on Automatic Control* **43**.

Chen, C.-T. (1999), *Linar Systems Theory and Design*, Oxford University Press.

Cornforth, W. & Croff, K. (2000), 'Development of an environment-sensitive navigation system for the auv autolycus', *Marine Technology* **37**, 238–245.

Demuth, G. & Springsteen, S. (1990), 'Obstacle avoidance using neural networks', *IEEE Proc Symposium Autonomous Underwater Vehicle Technology (AUV)* pp. 213–215.

Foss, B. A. (2004), 'Linear quadratic control', *Department of Engineering Cybernetics, NTNU* .

Fossen, T. (2002), *Marine Control Systems*, Marine Cybernetics AS.

Fossen, T. I. (1991), Nonlinear modeling and control og underwater vehicles, PhD thesis, Norwegian University of Science and Technology.

Fox, R., Garcia, A. & Nelson, M. (2000), 'A generic path planning strategy for autonomous vehicles', *The Univeristy of Texas-Pan American, Department of Computer Science* **Technical Report CS-00-25**.

Fryxell, D., Oliveira, P., Pascoal, A., Silvestre, C. & Kaminer, I. (1996), 'Navigation, guidance and control of auvs: An application to the marius vehicle.', *Control Engineering Practice* **CEP-4(3)**, 401–409.

Fujii, T. & Ura, T. (1996), 'Development of an autonomous underwater robot 'twin burger' for testing intelligent behaviours in realistic environments', *Autonomous Robots* **3**, 285–296.

Hart, P., J-Nilson & Raphael, B. (1968), 'A formal basis for the heuristic determintation of minimum cost paths', *Transaction on Systems* **Man and Cybernetics**, 100–107.

Healey, A. J. & Lienard, D. (1993), 'Multivariable sliding-mode control for autonomous diving and steering of unmanned underwater vehicles', *IEEE Journal of Oceanic Engineering* **18**.

Healey, A., Marco, D., McGhee, R., Brutzman, D. & Cristi, R. (1995), 'Evaluation of the nps phoenix autonomous underwater vehicle hybrid control system', *Proc American Control Conference* pp. 477–484l.

Horner, D., Healey, A. & Kragelund, A. (2005), 'Auv experiments in obstacle avoidance', *Oceans* **Proceedings of MTS/IEEE**.

Hyland, J. (1989), 'Optimal obstacle avoidance path planning for autonomous underwater vehicles', *IEEE Int Symposium on Unmanned, Untethered, Submersible Technology* pp. 226–278.

Hyland, J. (1990), 'A comparison of two obstacle avoidance path planning for autonomous underwater vehicles', *IEEE Proc. Symposium Autonomous Underwater Vehicle Technology (AUV)* .

Khalil, H. K. (2000), *Nonlinear Systems*, Prentice Hall Inc.

Knight, W. C., Pridham, R. G. & Kay, S. M. (1981), 'Digital signal processing for sonar', *Proceedings of the IEEE* **69**.

Kongsberg-Maritime (n.d.), 'Hugin 1000 - autonomous underwater vehicle'.
URL: *http://www.km.kongsberg.com*

Lane, D. & Trucco, E. (2000), 'Embedded sonar and video processing for auv application', *Proc Annual Offshore Technology Conference* **2**, 599–607.

Leonard, N. (1997), 'Stability of a bottom heavy underwater vehicle', *Automatica* **AUT-33(3)**, 331–346.

McKendrick, J. (1989), 'Autonomous knowledge-based navigation in an unknown two-dimensional environment eith convex polygon obstacles', *IEEE Int Symposium on Unmanned, Untethered, Submersible Technology* pp. 258–265.

Midtgård, ., Jalving, B. & Hagen, P. E. (n.d.), 'Initial design of anti-collision system for hugin auv', *FFI/RAPPORT -2006/01905 (In Confidense)* .

Nuno, P., Silvestre, C., Cunha, R. & Pascoal, A. (2006), 'A bottom-following preview controller for autonomous underwater vehicles', *Proc CDC2006 - 45th IEE Confernece on Decision and Control* .

Pascoal, A., Oliveira, P. & Silvestre, C. (1997), 'Marius: An autonomous underwater vehicle for coastal oceanography', *IEEE Robotics and Automation Magazine* **RAM-4(4)**, 46–59.

Ridao, P., Batlle, J. & Carreras, M. (2001), 'A new object oriented control architecture for autonomy. the reactive layer', *Control Engineering in Practice Journal* .

Rock, S., Wang, H. & Lee, M. (1995), 'Task-directed precision control of the mbari/stanford otter auv', *Proc of the Int Pogram Development in Undersea Robotics & Intelligent Control (URIC): A joint US/Portugal Workshop* pp. 131–138.

Sciavicco, L. & Siciliano, B. (2000), *Modelling and Control of Robot Manipulators*, McGraw Hill Inc.

Smith, F. & Ghidella, J. (2004), 'Incorporating m-code into simulink models', *Matlab Digest* **September**.

SNAME (1950), Nomeclatue for treating the motion of a submerged body through a fluid, *in* 'Technical and Research Bulletin NO.3-47'.

Tan, C., Sutton, R. & J.Chudley (2004*a*), 'Collision avoidance systems for autonomous underwater vehicles, part a: a reveiw of obstacle detection', *Journal of Marine Science and Environment* **C2**.

Tan, C., Sutton, R. & J.Chudley (2004*b*), 'Collision avoidance systems for autonomous underwater vehicles, part b: a reveiw of obstacle avoidance', *Journal of Marine Science and Environment* **C2**.

Warren, C. (1990), 'A technique for autonomous underwater vehicle route planning', *IEEE Journal of Oceanic Engineering* **15**, 199–204.

Winder, A. A. (1975), 'Ii. sonar systems technology', *IEEE Trans. Sonics Ultrasonics* **SU-22**.

www.ffi.no/hugin (n.d.).

Yoerger, D., Bradley, A. & Cormier, M. (2000), 'Fine-scale seafloor survey in rugged deep-ocean terrain with an autonomous robot', *IEEE Int Conference on Robotics and Automation* pp. 34–55.

Yuh, J. & Choi, S. (1999), 'Semi-autnonomous underwater vehicle for intervention missions', *Sea Technology Magazine* **Vol: 40, No: 10**, 31–40.

Zapata, R. & Lepinay, P. (1996), 'Collision avoidance and bottom following of a torpedo like auv', *IEEE Oceans Conference Record* **2**, 571–557.

Appendix A

User guide for the attached CD

The matlab and simulink files included on the CD are created using Matlab version 7.2.232 (R200a). If you are using older versions of matlab, you may experience problems with some or all of the files.

A.1 CD structure

The CD contains the following folders:

CAS simulations:

All of the Collision Avoidance system simulations mentioned in chapter 11 are placed her. Use the zoom buttons to zoom in and out on the plots and use the 3D rotate button to view the plot from different angles (recommended).

The complete simulator:

In order to run the complete model, the gnc toolbox is required. (This may be downloaded from: http://www.itk.ntnu.no/fag/gnc/matlab.htm) To run a simulation, DTM matrices for bottom and surface ice is required. To create an example landscape, run the file *ExampleTerrain.m*. Also controller gains for the autopilot must be calculated. To do this, run the file *LQtorque.m* Then simply run the simulation from the simulink file. Once the simulation is finished, use the matlab script *TrajectoryPlotter.m* to create a 3D plot of the simulation. To set a different WP path, double click on the 'Horizontal LOS Guidance' block and input desired Way points in the 'Waypoint North' and 'Waypoint East' boxes.

Matlab source code:

This folder contains all of the matlab files mentioned through the report.

Appendix B

Simulink diagrams

B.1 The nonlinear HUGIN model

huginmodel.mdl

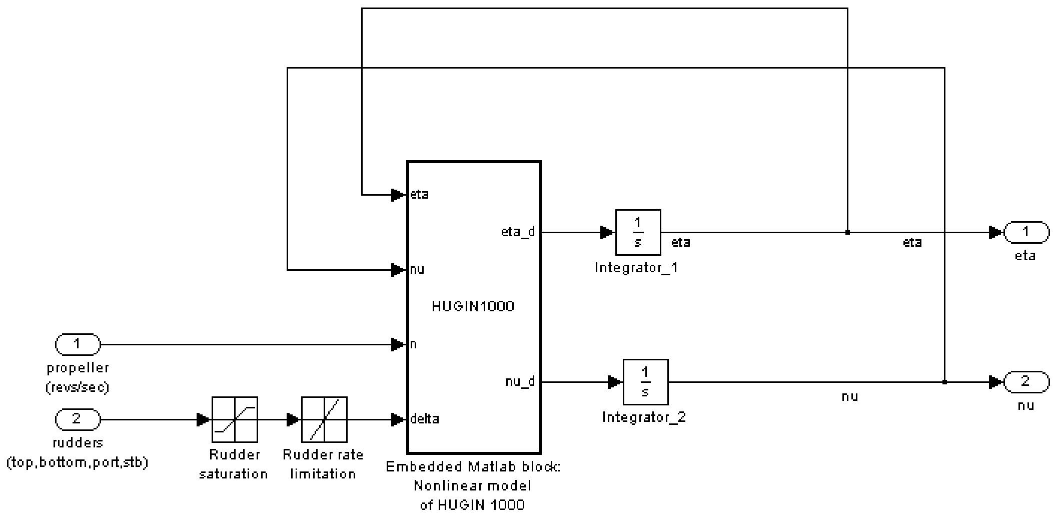


Figure B.1: The nonlinear HUGIN model implemented in Simulink

B.2 LQ torque control autopilot with rudder allocation algorithm

torqueAutpilot.mdl

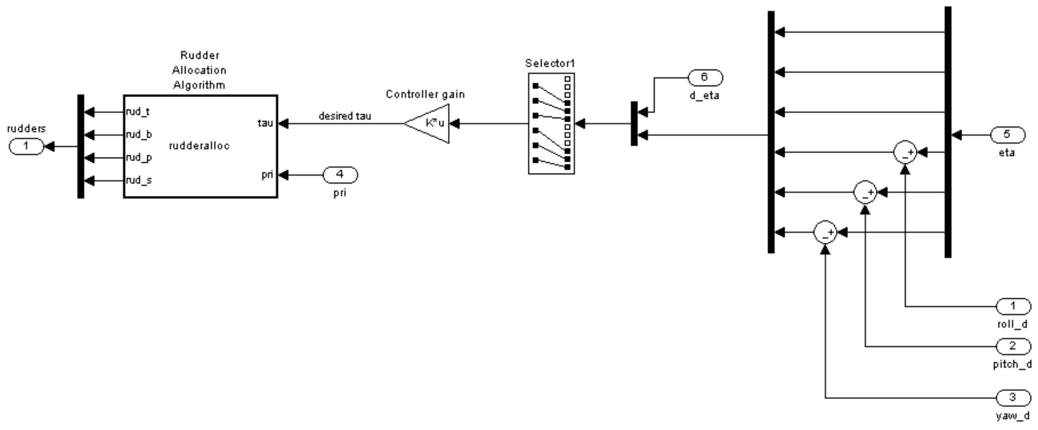


Figure B.2: The LQ torque controller with rudder allocation implemented in simulink

B.3 The complete system with CAS

simulator.mdl

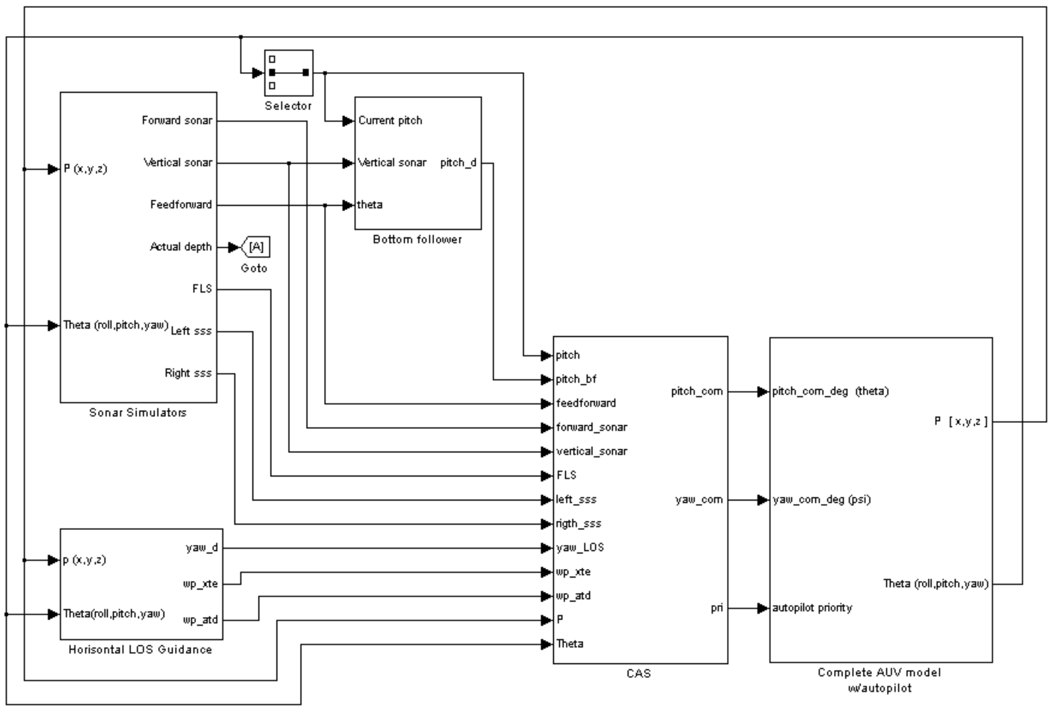


Figure B.3: The complete simulator with CAS implemented in simulink