Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Bachelor-Thesis

Design and Implementation of a Navigation Algorithm for an Autonomous Sailboat

Spring Term 2009

Supervised by:
Dr. Cédric Pradalier
Martin Rufli

Author:
Gion-Andri Büsser

Abstract

In the context of a focus project at the Federal Institute of Technology (ETH Zurich) which is concerned with the development of an autonomous sailing vessel, this bachelor thesis describes in detail the design and implementation of a path planner for Avalon, an autonomous sailing boat.

The planning algorithm is based on an A* search in 3 dimensions (*x, y, heading*) and takes into account current wind speed, wind direction and the nonholonomic properties of a sailing boat. The resulting paths minimize sailing duration in presence of static and dynamic obstacles. To break a world record in autonomous sailing, the algorithm will be used to guide Avalon across the Atlantic Ocean. In order to participate in the World Robotic Sailing Championship 2009 in Portugal, short course racing abilities are discussed, too.

Preface

I would like to use this preface to thank my two supervisors Cédric Pradalier and Martin Rufli, without whom this work would have not been possible. Cédric was very patient and extremely helpful when it came to the implementation of code and its subtlety. Martin I want to thank for all the constructive discussions and inputs that were the seed for a lot of ideas.

Additionally, I would like to thank my two good friends Hendrik and Jakob for their support throughout the project, their criticism and their thinking along. Thank you!

Zürich, in June 2009

Gion-Andri Büsser

Contents

Chapter 1

Introduction

1.1 Focus Project Avalon at ETH Zurich

The Federal Institute of Technology in Zurich enables mechanical engineering students in their third year of studies to participate in a focus project, a team work over the period of two semesters, with the goal of designing and developing a prototype from scratch to full operativeness. During the academic year of 2008 / 2009, one of the focus project teams is engaged in a venture to develop a completely autonomous sailing vessel for ultra long distances. The robustness and the durability of the boat AVALON will be proved in summer 2009, competing in the MICROTRANSAT 2009, an international competition to cross the Atlantic Ocean autonomously.

1.2 Thesis Overview

My bachelor thesis is part of that focus project and deals with the design and the implementation of a robust path planner for AVALON. The final goal is that the program will be able to generate a path from the current boat position to any given GPS coordinate, considering all the restrictions of a sailing boat's maneuverability and meeting all the requirements mentioned in section 1.3. The second important part discussed in this thesis will be dealing with the interface between the path planner and the controller.

The development of a controller for AVALON that follows the calculated course is not part of this work and is discussed in another bachelor thesis by Hendrik Erckens [1].

1.3 Path Planning Requirements

The MICROTRANSAT competition was initiated with the goal of crossing the North Atlantic Ocean autonomously. This task has never been faced and would - in case of accomplishment - mean to set a new world record. A tough environment like the Atlantic Ocean does not only make great demands on the mechanical parts of the boat, but it requires a software that can handle autonomous operation robustly for about two to three months and manage several possible risks. The basic goal is to develop a software structure that allows the boat to navigate on its own from the coast of Ireland to the Caribbean Islands. Below, the most important software requirements concerning navigation are listed:

Optimization for speed The program has to be laid out to find a path that always minimizes the time to sail towards the goal.

Nonholonomic Properties Due to its nonholonomic properties, every sailing vessel's maneuverability is restricted. The heading will always have to be calculated in respect to the current wind direction and wind speed.

Obstacles Besides having implemented all the coastal regions and islands as blocked zones, the software has to be able to avoid moving obstacles such as big ships.

Weather Forecast In order to avoid big storms on high sea or to sail into a region with not much wind, the use of weather forecasts has to be considered.

Robustness The path planner has to be constructed for autonomous long-term application, meaning that it has to be extremely robust and fail-safe.

1.4 Thesis Structure

The main aspects of this bachelor thesis were discussed in an IEEE paper that was submitted for publication for a special edition on Marine Robotic Systems of the IEEE Robotics and Automation Magazine. The paper is included in chapter 2. Chapter 3 will then go further into details of the “skipper” on board, the part that has control over all the decisions concerning navigation issues. The planner itself will the be illustrated in chapter 4, whereas section 5.3 elaborates the process of parameter optimization and demonstrates solutions with different cost-parameters.

1.5 General Terms

Following a short description and definition of terms that are used in this paper.

grid size Since a grid-based planning algorithm is used for the local planning (see section 4.3), GRID SIZE defines the distance between two parallel lines on the grid.

Chapter 2

Paper Submission to IEEE-RAM

Navigation Strategy and Trajectory Following Controller for an Autonomous Sailing Vessel

Hendrik Erckens (corresponding author), Gion-Andri Büsser,
Dr. Cédric Pradalier, Prof Dr. Roland Y. Siegwart
Autonomous Systems Lab, Swiss Federal Institute of Technology (ETH) Zurich
Leonhardstrasse 25, 8092 Zurich, Switzerland
E-mail: herckens@student.ethz.ch

***Abstract*—This paper describes the design and implementation of the autonomous sailing vessel AVALON. This boat, designed for participating in the MICROTRANSAT, will be able to autonomously cross the Atlantic Ocean. We present here a specially robust mechanical design and the navigation software that will plan optimal navigation course and efficiently control the boat on this course. The path planner uses an A* algorithm to generate the fastest path to a given destination. It is able to avoid both static and dynamic obstacles. By using a given polar diagram and measured wind data, it takes into account manoeuvrability constraints of a sailboat. The control system takes care of rudder and sail in order to follow a given heading while optimising speed. Additionally, the system is capable of automatically conducting manoeuvres such as tack and jibe. At the time of this writing, AVALON and all its software systems have been successfully tested more than a week with winds ranging from 0 up to 25 knots.**

I. INTRODUCTION

This paper introduces the autonomous sailboat AVALON as seen in figure 1. The general objective for building an autonomous sailing vessel is to further research and development in the area of unmanned, autonomous robotic vehicles that are exposed to heavy environmental conditions like the Atlantic Ocean. It has been established that there is a demand for autonomous sailboats to be used for ocean sampling and surveillance [1] but also for the implementation of this type of software in manned sailing vessels. Either by passively proposing optimal sailing routes or by actively supporting the sailor in dangerous situations by piloting the sail or rudders, an integrated autonomous sailing system could back up and facilitate a lot of situations.

A. Our Goals and Objectives for this Work

Our boat AVALON was designed to compete in the MICROTRANSAT Challenge and thus has to withstand the harsh conditions on the Atlantic Ocean.

The goal of this paper is to present three aspects of AVALON: its mechanical and electrical design, our implementation of a software that is capable of calculating the optimal path to reach a given destination as quickly as possible, and the design of a controller able to safely and efficiently guide the sailboat to its destination, taking into account the current environmental conditions, such as wind speed and wave height.

B. Review of Existing Autonomous Sailing Boats

An autonomous sailboat has to be capable of sailing without human intervention. That means, that all tasks that are normally carried out by the sailor on a conventional sailboat have to be done autonomously. That involves navigation as well as working the rudder and sail in order to steer the defined course. Other autonomous sailboats have been developed mostly by MICROTRANSAT participants [2], [3], [4], [5], but also commercially usable products [6] have been built.

II. A BOAT DESIGNED FOR SURVIVAL

This section summarises some details about the main components of AVALON. Table I shows the major technical details of AVALON.

A. Rig

The rigging system is one of the most important parts of the whole assembly. A defect in its structure will inevitably

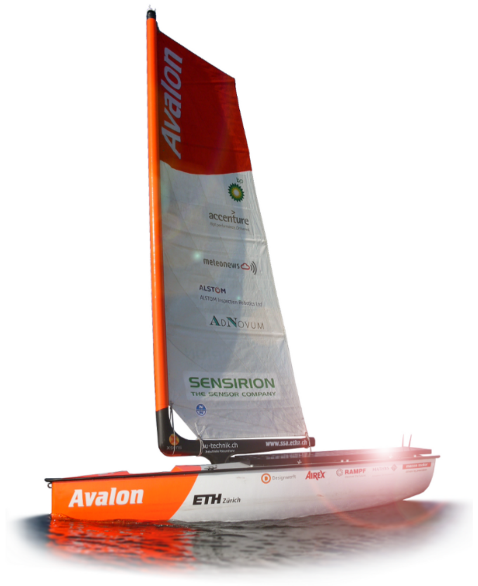


Fig. 1. Avalon ready to sail on the lake of Zürich

cause the whole project to fail. The following aspects were considered for the design:

- High loads and forces on the mechanical structures due to strong winds and heavy weather conditions.
- Highest demands on reliability. There is no chance to repair anything throughout the journey.
- Preferably efficient force transmission from the actuator to the sail in order to save as much energy as possible

During the design phase it turned out that a balanced rig prevails over a conventional rig.

The balanced rig is much more efficient regarding energy consumption than any other conventional rig. The force needed to position the sail is reduced by over 50% due to the balanced distribution of the sail load. Furthermore, AVALON's rig does not use any ropes that could generate knots and jams. The rig is pivoted on a central bearing without shrouds and stays. The result is a simple and reliable construction.

B. Hull

The hull design is based on form parameters, e.g. length, draft and beam. Parameters were used as input data for B-spline curves and surfaces. Instead of modifying those curves and surfaces, we used constrained optimisation algorithms to generate the geometry. The algorithm minimises a target function, which is defined so that a certain combination of parameters results in a desired hull shape.

TABLE I
TECHNICAL DATA OF AVALON

Variable	*Description*	*Value*
L_{OA}	Length over all	$3.95\,m$
B_{max}	Maximal beam	$0.7\,m$
T	Draft without keel	$0.25\,m$
D_{max}	Draft over all	$2\,m$
H_{rigg}	Height of the rigg	$5.7\,m$
A_{sail}	Total sail area	$8.4\,m^2$
∇	Submerged volume	$0.44\,m^3$

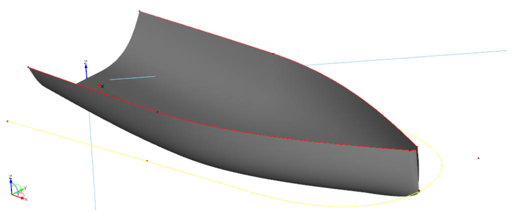


Fig. 2. Hull

The final hull was laminated inside a female mold using glass fibre sandwich. Epoxide resin was sucked into dry glass fibre material by vacuum in a so called *Infusion Process*. Compared to conventional laminating, this method is much cleaner and more convenient.

C. Keel and Rudder

1) Keel: In order to achieve a sufficient righting moment and stability during heavy weather situations, AVALON's keel with a draft of two meters consists of a slim fin with a 160 kg ballast bulb.

The fin and parts of the bulb were made of high-modulus carbon fibre in precisely milled polyurethane female molds. After hardening and tempering, the bulb was filled with lead and the two halves were glued together.

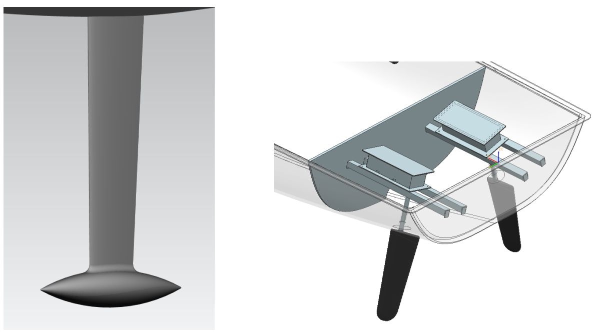


Fig. 3. Keel with fin and ballast bulb (left), Twin rudder system (right)

2) Rudder: A twin-rudder system was selected for AVALON to make sure to have sufficient steering effect in every sailing situation. Angular mounted twin-rudders provide better control at high heeling angles compared to single rudders.

Assembled inside the hull, the rudder actuators are well sealed and protected against water and humidity.

D. Power Supply

The main power supply is realised through two square meters of solar panels providing a total of 360 Wp. The collected energy is stored in four lithium-manganese batteries. Each battery consists of 70 single cells and has a capacity of 600 Wh at a nominal voltage of 25.2 volts. Lithium-manganese batteries were chosen mainly because of their weight but also because they are fairly safe to use.

For back-up power, the boat has a direct-methanol fuel cell on board. This fuel cell is automatically activated when the voltage drops under a certain value, the *switch on* voltage. It then charges the batteries until the *switch off* voltage is reached. In theory, the solar cells provide enough power for the boat's systems. The fuel cell only serves in case of enduring bad weather or other unforeseeable circumstances.

III. EMBEDDED SYSTEM DESIGN

A. Main Computer

The brain of the system consists of different hardware components located in the centre of the hull. The main computer as well as different peripheral devices such as a serial-to-USB adaptor, satellite modem and a wireless LAN hub are in a double sealed fibre glass box.

The main computer MPC21 from DIGITALLOGIC is a 500 MHz device with 1024 MB RAM and a compact flash hard-drive. The device has an average power consumption of

Fig. 4. Main PC and brain of AVALON [7]

about 8 watts, the protection of a metal housing and a total weight of less than 1 kg.

B. Sensors

All desired information such as position, heading or speed are measured by several sensors located all over the boat. To get a fully controllable system, data is collected from the following sensors:

a) GPS & IMU: The *Inertial Measurement Unit* provides all information about velocity and acceleration in all 6 degrees of freedom. Combined with a built in GPS-receiver, it also determines the boat's position with great precision.

b) Wind: Mounted on top of the mast, the wind-sensor provides wind speed and direction. AVALON has an ultrasonic wind-sensor that promises less failure than a conventional sensor with a turning wheel.

c) AIS: This Sensor receives data such as position and velocity from other boats via VHF. The AIS system is an additional means of perception that ensures that collisions with large commercial ships are avoided.

C. Satellite Connection

In order to guarantee a safe connection to the boat during the transatlantic journey, a very reliable communication system is needed. It will not only provide periodic position and status reports, but also the possibility to interact with the boat in case of an emergency. All these tasks can only be accomplished using a satellite network. Currently, the only two satellite networks that provide coverage on the Atlantic Ocean are IRIDIUM and INMARSAT. The choice between these two options was based on the available hardware for each of these networks. The IRIDIUM 9522-B modem, which was finally chosen, has a very flexible interface. As IRIDIUM also acts as an Internet Service Provider (ISP), there is the possibility to establish a *Secure Shell* (SSH) connection to the boat in case of a very severe malfunction of the software.

D. Software

All software is developed in C++ and Python and runs on a Linux OS. The base of the entire software structure is the middleware DDX [8]. This software manages a shared memory called the store and thus provides a means of communication between the individual programs (see figure 5). Thanks to DDX, the control system is structured in closed program parts which are connected to the store by reading and writing global variables. For instance, there are sensor drivers which read out actual sensor data from the hardware and write it directly to the store. Another program requiring sensor data can now read needed data from the store. A distinct advantage of DDX is the modular software structure. If ever a program crashes, it can be restarted without interrupting the other processes. The different subprograms are depicted in figure 5.

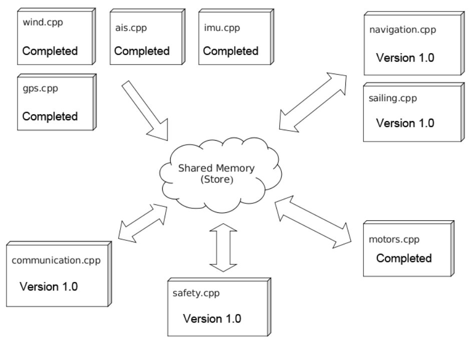


Fig. 5. Software organisation

IV. AUTONOMOUS SAILOR

A. Objectives

This section describes the design of a controller that is capable of robustly steering a sailboat along a predefined course while always setting the sail to the optimal angle to the wind in order to generate as much driving force as possible. For that, it also has to be able to conduct manoeuvres such as tack and jibe. At the same time, the controller should take into account the changing environmental conditions, such as wind speed and direction and react accordingly. Furthermore, since the navigator (section V) calculates a global trajectory and relies on the control system to follow this course, the autonomous sailor has to compensate for drift.

B. Existing Works

There are various systems available that assist a sailor in steering a sailboat. Most of these are commercially available autopilots, that operate either with a built in magnetic compass or with a wind vane [9]. These systems are finished products, that have been used for many years and are very robust and reliable. We did not use commercial products, mainly because of interfacing issues. Autopilots are designed for a human sailor on board who does the navigation and has to tune the sails. Although some of the systems are even able to perform tacks and jibes, the manoeuvre has to be confirmed by the human sailor pressing a button on the autopilot's control panel. Since the software integrated in commercial autopilots is typically closed source, this problem cannot be solved by simply reprogramming the device. Furthermore, most autopilots only control the rudder, so we would still have to figure out a way to tune the sail.

Apart from the commercial products, there are also several scientific works regarding control systems for autonomous sailboats, most of them also by participants in the MICROTRANSAT. Developments reach from a fuzzy logic controller for both rudder and sail [10], over an LQG controller based on a non-linear 3 degrees of freedom (DOF) model [11], up to a complex self learning AI system that has been successfully tested on an Open60 racing yacht. Our work is mainly influenced by Briere [2], who describes a simple controller based on a state machine.

C. System Modelling

In order to design a robust controller and identify parameters in a controlled environment, a simulator was needed. Although there are several sailing simulators available, most of them are made for gaming. Rather than having an interface that allows them to be steered by a self-developed control system, they are meant to be operated by hand. For this reason we implemented a simulator in Matlab/Simulink.

To gain a general system that could later be extended with boat/wave interaction, the boat was modelled as a rigid body with full 6 DOF. The system has 14 state variables: 3 for position, 3 for attitude, 3 for velocity, 3 for turning rates plus rudder and sail angle. Inputs to the plant are rudder angle γ_{rudder}, sail angle γ_{sail}, true wind speed $v_{\text{wind\_true}}$ and direction Ψ_{wind}, while outputs are the boat's position and attitude and their corresponding derivatives.

Forces were introduced at various points of the boat to model the interactions between water and hull as well as wind and sail.

1) Sail Force: In order to calculate the force generated by the sail, the apparent wind angle is needed which is computed in a vector operation from the true wind angle and the boat's velocity. The 2 components of the sail force F_{sail} are then modelled using the standard approximation for air foils in a moving fluid

$$\begin{aligned} F_{\text{sail\_lift}} &= \frac{1}{2} \cdot \rho_{\text{air}} \cdot v_{\text{wind\_app}}^2 \cdot A_{\text{sail}} \cdot c_l(\alpha_{\text{sail}}) \\ F_{\text{sail\_drag}} &= \frac{1}{2} \cdot \rho_{\text{air}} \cdot v_{\text{wind\_app}}^2 \cdot A_{\text{sail}} \cdot c_d(\alpha_{\text{sail}}) \end{aligned} \tag{1}$$

where ρ_{air} is the density of air, A_{sail} is the apparent area of the sail and c_l is the lift coefficient which depends on the sail's shape and the angle of attack α_{sail}.

2) Rudder Force: The rudder force is also modelled using equation 1. Since the rudders are at the stern of the boat, the force induces a moment around the vertical axis and thus affects the boat's heading.

3) Resistances: There are several damping forces that depend on the velocity and rate of turn of the boat. Those are mainly the resistance of hull and keel in all 3 axes of translational freedom and the damping of rotations. These are modelled using the equation

$$F = d \cdot v^2 \tag{2}$$

where v is the velocity or rate of turn and d is a parameter that has to be identified in experiments.

D. Controller Principle

In our control system layout, rudder and sail are controlled in two separate Single Input Single Output (SISO) systems that are assumed to be independent of each other. During testing, this assumption has proved to be very reasonable.

1) Sail Control: For the sail, the controlled value is the angle of attack. This directly depends on the sail's angle. Since AVALON's sail and rudder motors already come with a built in positioning controller, this subsystem is sufficiently controlled by just writing a reference value to the motor controller.

This reference value is derived from a predefined optimal Angle Of Attack (AOA) that was found in experiments. However, for safety reasons, the wanted AOA also changes dynamically depending on the wind speed. Basically, as the wind speed increases, the wanted AOA approaches zero, which reduces the sail force generated by the wind. With this behaviour, the boat remains steerable even in strong winds.

2) Rudder Control: The second subsystem controls the boat's heading using the rudder as input. Since the heading changes dynamically and is not directly dependent on the rudder angle, this system takes a little more consideration. For this purpose, a PID controller was designed in an iterative process using the Matlab simulation. It was then tested and optimised in the real boat (see section IV-E).

In order to compensate for leeway drift, the boat has to sail somewhat closer to the wind than the desired heading DH_{nav} calculated by the navigator. Since drift changes considerably with the wind and wave conditions, it is important to know the current drift in order to compensate it. AVALON is equipped with an Inertial Measurement Unit (IMU) with a built in GPS receiver. With this sensor, the drift speed v_{drift} can be accurately measured at any time. From the drift, a new desired heading $\text{DH}_{\text{sailor}}$ is calculated using the equation

$$\text{DH}_{\text{sailor}} = \text{DH}_{\text{nav}} + \arctan\left(\frac{v_{\text{drift}}}{v_{\text{boat}}}\right) \tag{3}$$

3) State Machine: A sailboat can not sail at all angles to the wind. Figure 6 shows the ranges that we cannot sail in gray. In order to sail as fast as possible, it has to be differentiated between different types of sailing. To do that, the controller is designed as a state machine that switches states depending on the angle to the wind. The 5 states are depicted in figure 6.

4) Normal Sailing: If the desired heading given by the navigator (section V) is in the sailable (white) range, the system is in state `Normal Sailing`. In this state the rudder controller follows the given desired heading, while the sail controller permanently adjusts the sail to achieve the optimal angle of attack on the sail. If the desired heading changes within the sailable range, the rudder controller simply follows the new course while the sail controller maintains a constant angle of attack.

5) Upwind Sailing: If the desired heading for whatever reason is set in the not sailable (gray) range, it cannot be directly followed. The objective in this case is to make as much velocity towards the wind as possible. In `Upwind Sailing` mode the sail is set to the tightest position that is reasonably possible. While the sail angle is kept at this constant position,

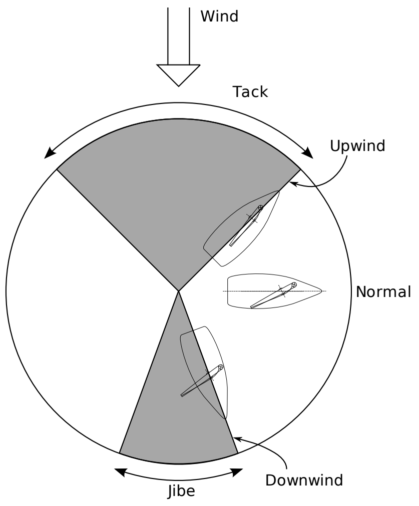


Fig. 6. Possible courses of a sailing yacht with respect to the wind and corresponding control system states. The range with wind directly from behind could be sailed in theory, but is not very efficient and rather unstable.

the rudder controller now keeps a constant angle to the true wind and thus takes advantage of all wind shifts.

E. Test Results and Analysis

Several tests were undertaken to verify and optimise the control system. Before any testing on the water was done, the state machine's transition conditions had to be verified. This was done by putting the boat on a trailer on a sufficiently windy day and turning it around its vertical axis while constantly checking the current state. We encountered many problems that were caused by the sign change between -180^o and $+180^o$. Some time could have been saved, if the whole control program, including the state machine, had been tested more thoroughly in the simulation environment.

The controller was then tested for its ability to keep a desired heading and to react to changes in desired heading. After some tuning of the PID parameters we found that the parameters in table II yield fast reactions without too much overshoot. A step response in winds of about 20 knots is illustrated in figure 7. Note that 20 knots is already a strong wind and that in lower wind speeds we experienced much less oscillation around the reference value. We also applied disturbances to the boat by deviating it from its course by hand. The reaction was the same as with changes in desired heading.

TABLE II
PID PARAMETERS

P	0.3
I	0.005
D	30

Figure 8 shows current heading, desired heading and wind direction during three tacks. Between the tacks, the controller state machine is in state `Upwind Sailing`. It can be very well seen, that the boat keeps a constant angle to the wind and takes advantage of wind shifts as expected. A lot of time was invested into correct timing during tack and jibe. In the beginning we had several problems that resulted in inappropriate sail tuning during or right after manoeuvres. This sometimes even resulted in the boat sailing backwards.

Figure 9 shows how the boat follows a trajectory that was calculated by the navigator. The navigator's calculation was started at the bottom left where the dashed line begins. After the calculation was completed, the boat jibes and sets the new course. After a few seconds the drift compensation starts to work and the boat's real course approaches the desired trajectory.

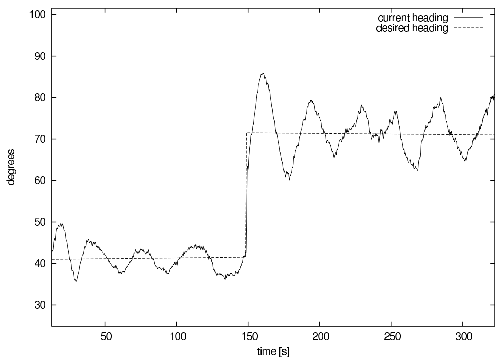


Fig. 7. Step response in state `Normal Sailing`. Conditions were 20 kn wind from 140°

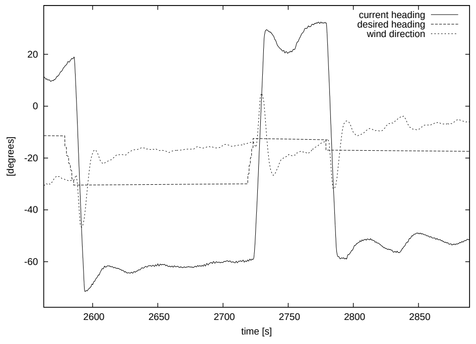


Fig. 8. 3 tacks, with `Upwind Sailing` in between.

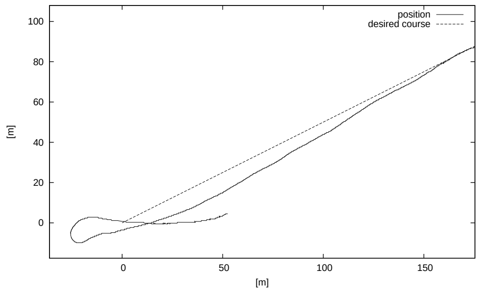


Fig. 9. GPS plot of the controller following a desired trajectory calculated by the navigator

V. NAVIGATION STRATEGY

A. *Principle of motion planning for a sailing boat*

Due to the nature of every sailing vessel, AVALON's manoeuvrability is limited in relation to wind direction. Additionally, due to the non-holonomic properties of a sailing vessel, it is not possible to change the heading on the spot immediately. The trajectory tracker (see section V-G) has to keep that in mind and operate in a way that allows a certain space to make heading changes. For the implementation of a navigation algorithm, these constraints imply that using accurate wind information for every calculation is essential.

B. *Literature Research*

Path planning in the naval industry is not a new topic. For many years, commercial ships have used weather forecasts to generate an optimal path with respect to speed or bypassing bad weather situations. For sailing ships, commercial solutions by companies like B & G or NORTHSTAR exist, but do not offer any possibility to be integrated into our embedded system. Looking at different navigation and routing strategies, two fundamentally different ideas emerge: Roland Stelzer [3], who has published his approach also for implementation in an autonomous sailing boat, projects the speed vector on the target direction and tries to find a sailable heading that maximises that projected speed vector. Although in small scale it might find a good and fast path, this approach is not sufficient when it comes to larger search areas with obstacles and additional inputs like weather forecasts.

The other approach is to use grid based path planning algorithms that allow us to generate optimal solutions considering the path from start to goal and enable the integration of specific constraints (see section V-A).

Breadth first method and *Depth first* are both based on a very simple principle and serve as archetypes for many important algorithms [12], but may result in very long running time. *Dijkstra* on the other hand is a directed search [13] and guarantees optimal results. A* is a modified version of *Dijkstra* that also considers the heuristic distance estimate of the path from current position to the destination [14], which makes it more directed than *Dijkstra*. One of the problems of grid-based planning algorithms is that the angles are artificially constrained. The Theta*-Algorithm [15] deals with that handicap and produces more realistic looking paths than A* or even smoothed A*. A further extension of A* is D* [16], which incorporates dynamic changes (i. e. moving obstacles) along the path and does not require to replan the whole path from current position to destination anymore.

The proposed approach uses a simple A* algorithm which is modified to comply with all the sailing constraints. It must also be noted that there is few computation time constraints for a sailing boat crossing the Atlantic ocean. Even if path planning requires a handful of minutes, this is still an acceptable behaviour.

C. *Planning Strategy*

Considering the distance of about 4'200 nautical miles AVALON has to sail on its journey across the Atlantic Ocean and its slow speed of about 4 to 5 knots[1], it is essential to distinguish between a global and a local planning. Global planning will be done by fixing way-points across the Atlantic based on expert knowledge, weather and current analysis. The local planner will then plan from current position to the next global way-point using wind data from the on board sensor. The planning algorithm will be triggered and controlled by a trajectory tracker (section V-G), which knows exactly towards which way-point AVALON is sailing at the moment.

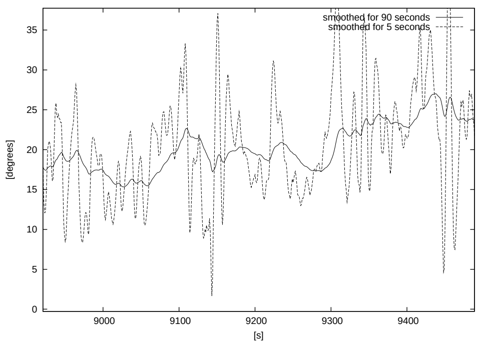


Fig. 10. Differently filtered wind information

D. *Interfaces with the autonomous sailor*

An important part of the software structure is the interface between the navigation program and the sailor (see section IV). The main reason why we decided to pass *headings* instead of way-points from the trajectory tracker to the sailor, was that the navigator calculates an optimal path with respect to path duration for specific angles to the wind. Drift that pushes the vessel off the trajectory will be handled by the sailor, a new calculation will only take place after the boat has a certain deviation from the set trajectory (see section V-G).

1) Wind data: The wind the local path planner uses is an average of the measurements over the last 90 seconds. The sailor will then follow the received desired heading using a much more accurate wind direction that was averaged only over a period of 5 seconds (see section IV). Figure 10 shows a comparison of the two different wind sets.

E. *Planning algorithm*

As explained earlier, *Dijkstra*'s graph search is the basic algorithm used to perform the path search. To be able to allocate costs for heading changes, we work in 3 dimensions, with x and y for the geographic position and θ for the heading of the boat at that position. To get sailable and reasonable results, the following costs have been implemented additionally:

- Sailing-Time from start node
- Estimated sailing time to the end node (heuristic)
- Cost for heading changes
- Cost for manoeuvres (tack and jibe)
- Cost for offset from the straight connection between start and end node (tunnel cost).

[1] 1 knot is 1 nautical mile per hour

1) Grid compatible coordinate system: To represent GPS points on a map, a coordinate transformation is necessary. Since we plan only for short distances ahead of us (local planning), we can use a very simplified transformation into meter coordinates

$$x = R_E \cdot \cos(\text{lat}) \cdot \frac{\pi}{180\,deg} \cdot \text{lon}$$
$$y = R_E \cdot \frac{\pi}{180\,deg} \cdot \text{lat} \tag{4}$$

where R_E is the earth radius, lon and lat the GPS coordinates in degrees. This transformation assumes a flat water surface [3]. In order to initialise the local map as small as possible, the algorithm places the origin as close as possible to either start or target coordinate.

2) Polar diagram: In order to calculate the boat speed for a specific angle to the wind, a polar diagram[2] of AVALON is needed. Since creating a detailed diagram for our boat would require constant wind and wave conditions over a long time, we created a polar diagram based on existing diagrams for bigger vessels. Testing has shown that we get reasonable results by restricting the sailable *angle to the wind* to the area between 45^o and 160^o (see figure 6).

3) Heading: In order to limit the size of the 3D grid used in this paper, we use a discretisation of the possible boat heading into 16 possible headings. In usual 2D grid-based path planning, transition from one cell to its 8 neighbours is considered. This results in only 8 possible changes of orientation. By considering a neighbourhood of 24 cells (8 cells around the starting cell, plus 16 cells one cell further), we can reach 16 different heading changes in one transition. This leads to smoother paths and smaller number of way-points on the final path.

We further reduce the computational complexity of the planner by considering that the maximum change of heading in one transition is 90^o. This restriction is consistent with the experience of human skippers.

F. Cost Factors

1) General Costs: The general aim of this algorithm is to find the shortest path for a sailing vessel to go from point A to point B. The general cost we give for every movement is therefore *sailing time to the next node*, calculated by equation 5. The speed solely depends on the angle to the wind and the wind speed.

$$\text{sailing time} = \frac{\text{distance to next node}}{\text{speed}} \tag{5}$$

2) Heuristic Estimate: The heuristic estimate is used to evaluate the remaining time to the goal. According to [13], the only constraint on the heuristic is that it has to overestimate the cost of the remaining path. For simplicity sake, the heuristic returns the time it would take to sail to the goal at a very conservative speed of 1 knot. More complicated options have been considered, but none of them provided well defined behaviour in all situations.

[2]In boating term, the polar diagram gives a boat velocity as a function of its heading to the wind and the wind velocity

3) Turning Cost: To generate paths as smooth as possible, the algorithm places costs C_{new} for heading changes, using a simple linear factor f in equation

$$C_{\text{new}} = f \cdot |\theta_{\text{curr}} - \theta_{\text{new}}| \tag{6}$$

where θ_{curr} and θ_{new} are the respective headings.

4) Cost for Tack or Jibe: By giving a certain cost for tack or jibe, we can indirectly control the number of manoeuvres AVALON sails. The principle is very simple, we check if the sign of $(\theta_{\text{curr}} - \text{winddirection})$ and $(\theta_{\text{next}} - \text{winddirection})$ is the same. If it is different, an additional cost will be added to the total cost.

5) Tunnel cost: In order to favour paths that stay closer to the straight line path, we allocate costs increasing with the distance from the direct connection of start and target node. It has proved best to increase the cost only little for the areas in the middle of the tunnel but increase it abruptly when closing in on the borders.

6) Obstacle avoidance: To avoid islands and coastal regions, the algorithm gives extremely high costs for passing prohibited terrain. To locate these areas, it parses information from a map.
To avoid commercial ships on the Atlantic, AVALON can receive AIS signals from most ships travelling over the Atlantic. Knowing ship speed, direction and current position, our algorithm will place no-go areas around these ships to make sure to bypass them.

G. Trajectory tracker

The trajectory tracker has mainly two important tasks:

- Triggering the local planner
- Calculating a desired heading for the sailor

1) State machine: To manage the different tasks, the trajectory tracker can switch between different states:

a) Standard Navigation: If not approaching a target or buoy, this state generates headings for the sailor, going from way-point to way-point. If the wind changes more than 20° or if the ship distances itself from the current trajectory for more than a predefined distance, a new calculation will be initiated.

b) Goal Approach: If approaching a target position, the trajectory tracker will always send a heading that is directing exactly towards that point.

c) Buoy Approach: This is a state designed especially for short course racing. Due to regulations that always require a port-side surrounding of the regatta buoys, this state will - when approaching a buoy - always write a heading that will lead the boat around the buoy.

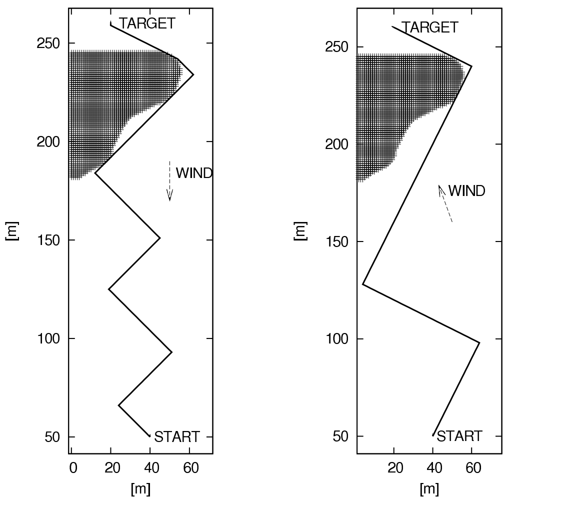
d) New Calculation: In this state, the trajectory tracker manages and supervises the new calculation. After having checked that the new way-points have been stored, it switches back to *Standard Navigation*

H. Tests and Discussion

1) Testing environment: Most tests and parameter optimisation was done by visualising the generated path on screen and discussing its quality. Figure 11 shows two calculated path examples, both having to pass an obstacle and dealing with difficult wind direction like upwind and downwind. Well apparent is the restrictive tunnel cost which forces the boat to stay along the direct connection of *start* and *target* node.

Another important and not yet solved problem is the final heading at the target node that has to be known in order for *Dijkstra* to start the search. At the moment the algorithm chooses a final heading that is as close as possible to the direct heading of a straight line from start to destination but always differs at minimum 45^o from the wind direction. This usually produces good results but it does not take obstacles into account which means that sometimes an additional, unnecessary turn is produced.

In practice, this issue will be solved by starting planning a new path before completing the execution of the current one.



(a) Upwind (b) Downwind

Fig. 11. Path examples for upwind and downwind sailing with obstacles

Fig. 12. Avalon during testing in high winds

VI. Conclusion and Outlook

The proposed navigation and control software has been successfully implemented and tested in our autonomous sailboat Avalon. The resulting system is robust and is capable of reaching any given point while avoiding collision with obstacles. It has been tested in conditions ranging from almost 0 to 25 knots of wind speed. In very low wind speeds it was seen that the boat is able to sail much better by itself than steered by us via remote control, because it is always aware of the exact wind direction, even when human sailors hardly notice any wind at all.

Implementing a conventional and proved path planner like Dijkstra's for our navigation system has proved well. Visually and analytically evaluated results show that we achieve better solutions by taking the whole path into account than by always optimising the heading at every current position. However, by adding additional costs (like tunnel cost), optimal path solutions are not always guaranteed.

Our goal for future work is to further optimise the software, especially regarding energy consumption. For example a hysteresis could be added to the sail tuner so that the sail does not constantly adapt to minor changes in wind direction and speed. A further area of improvement could be in making use of weather forecasts in the global navigation. For our boat this does not add much value, because it is too slow to react to forecast weather changes.

References

[1] N. Cruz and J. Alves, "Autonomous sailboats: an emerging technology for ocean sampling and surveillance."

[2] Y. Briere, "IBOAT: An autonomous robot for long-term offshore operation," in *Electrotechnical Conference, 2008. MELECON 2008. The 14th IEEE Mediterranean*, 2008, pp. 323–329.

[3] R. Stelzer and T. Pröll, "Autonomous sailboat navigation for short course racing," *Robotics and Autonomous Systems*, vol. 56, no. 7, pp. 604–614, 2008.

[4] J. Alves and N. Cruz, "FASt-An autonomous sailing platform for oceanographic missions."

[5] C. Sauze and M. Neal, "Design Considerations for Sailing Robots Performing Long Term Autonomous Oceanography," in *Proceedings of The International Robotic Sailing Conference, 23rd-24th May*, 2008, pp. 21–29.

[6] May 2009, `http://www.harborwingtech.com`.

[7] 2009, `http://www.digitallogic.com`.

[8] P. Corke, P. Sikka, J. Roberts, and E. Duff, "Ddx: A distributed software architecture for robotic systems," in *Proc. Australian Conf. Robotics and Automation*, Canberra, December 2004.

[9] D. Hochseeportverband *et al.*, *Seemannschaft*. Delius, Klasing, 2008.

[10] R. Stelzer, T. Proll, and R. John, "Fuzzy logic control system for autonomous sailboats," July 2007, pp. 1–6.

[11] G. Elkaim and R. Kelbley, "Station Keeping and Segmented Trajectory Control of a Wind-Propelled Autonomous Catamaran," in *Proceedings of the Conference on Decision and Control*.

[12] C. Leiserson, R. Rivest, T. Cormen, and C. Stein, *Introduction to algorithms*. MIT press, 2001.

[13] S. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[14] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, Issue 2, pp. 100–107, July 1968.

[15] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta^*: Any-Angle Path Planning on Grids," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 22, no. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007, p. 1177.

[16] A. Stentz, "The Focussed D^* Algorithm for Real-Time Replanning," in *International Joint Conference on Artificial Intelligence*, vol. 14. LAWRENCE ERLBAUM ASSOCIATES LTD, 1995, pp. 1652–1659.

Chapter 3

Skipper

As written in chapter 2, section V.G, the skipper overlooks and controls the whole navigation software. Its two main tasks are to trigger the path planner to generate new waypoints and to pass a currently desired heading to the controller of the boat. Figure 3.1 shows an overview over the two tasks. Realized with a state machine, the skipper can switch between four different states: `standard mode`, `new calculation mode`, `buoy approach mode` or `target approach mode`.

3.1 Sailor Interface

Concerning the interface between SKIPPER and SAILOR (controller), we had several possibilities. We could either pass waypoints, headings or specific headings to the wind. As mentioned earlier, we eventually decided to pass headings. For further illustration, figure 3.3 depicts a standard situation at sea with 3 waypoints. The heading that is passed to the controller is always the exact heading from the previous waypoint ($\mathrm{WYP}_{\mathrm{prev}}$) to the current waypoint ($\mathrm{WYP}_{\mathrm{curr}}$), not a heading always pointing at the current waypoint ($\mathrm{WYP}_{\mathrm{curr}}$). The main reason for the decision was that the planning was laid out to be the best option with respect to *angle to the wind.*

Always passing direct headings from the boat to the current waypoint ($\mathrm{WYP}_{\mathrm{curr}}$) would force direct piloting of every waypoint which is neither optimal nor necessary for long paths. By passing headings to the wind, following a certain trajectory would get very hard since the sailed heading would fluctuate extremely.

3.1.1 Wind Data

In order to generate a path, we need wind data that can be assumed constant over time. A good choice for filtering the wind sensor data over time was 90 seconds. The controller that has to follow the given heading as precisely as possible then uses a more accurate wind direction (filtered only over the last 5 seconds) to set the sail and the rudder. Figure 3.2 contrasts the two wind data sets.

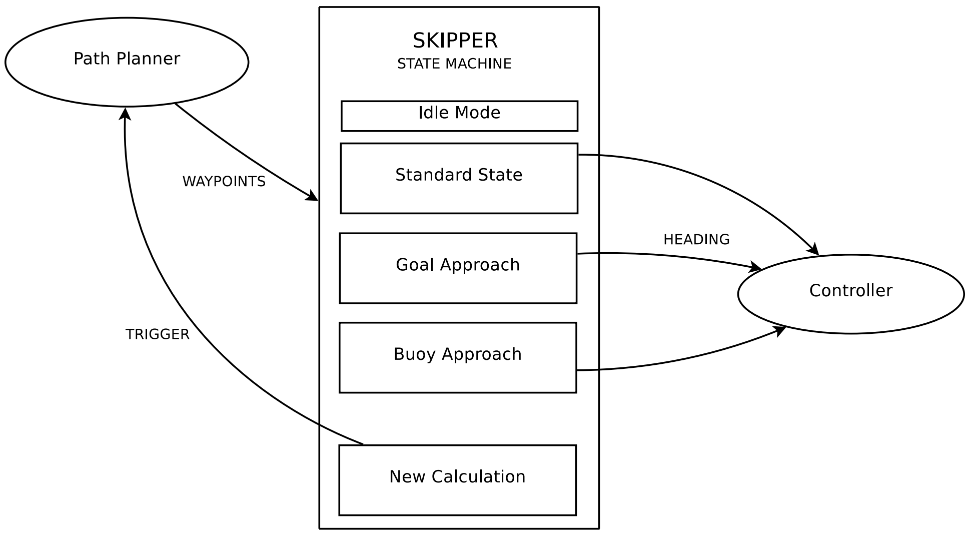


Figure 3.1: Schematic representation of the skipper state machine

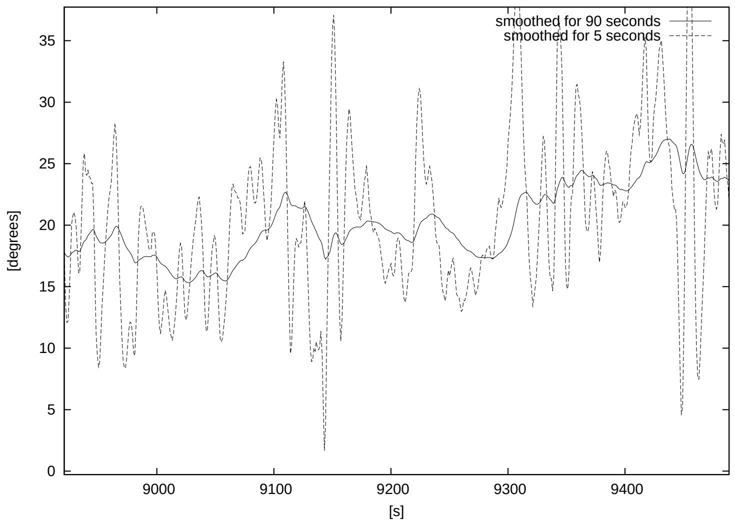


Figure 3.2: Differently sampled wind data from the Vierwaldstättersee

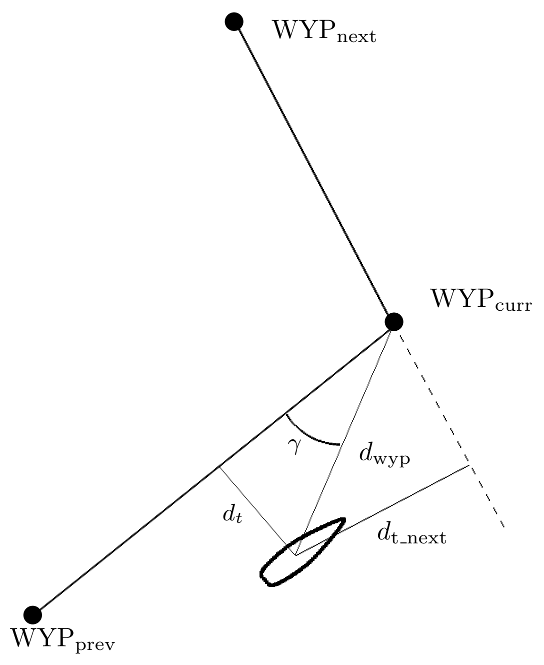

$\mathrm{WYP}_{\mathrm{next}}$
$\mathrm{WYP}_{\mathrm{curr}}$
γ
d_{wyp}
d_t
$d_{\mathrm{t\_next}}$
$\mathrm{WYP}_{\mathrm{prev}}$

Figure 3.3: Standard situation at sea

3.2 States

This section shortly describes all the different states the skipper can be in and highlights the standard mode as an archetype for all the modes. Figure 3.3 shows a general overview of a standard situation at sea. The angle γ as well as the distance to the current trajectory d_{t}, the distance to the next trajectory $d_{\mathrm{t\_next}}$ and the distance to the current waypoint d_{wyp} can be calculated and are used for decision making by the skipper.

3.2.1 Standard Mode

There are two significant actions that can be triggered in that state:

- Head for the next waypoint
- Activate a new calculation by switching into the mode `new calculation`

In addition, the desired heading for the controller is updated if necessary.

New Waypoint Two special conditions allow the skipper to advance one waypoint, meaning that $\mathrm{WYP}_{\mathrm{next}}$ becomes $\mathrm{WYP}_{\mathrm{curr}}$. Both conditions that can trigger that action are "double conditions", meaning that they can only become true, if the distance to the current waypoint (d_{wyp}) is smaller than the GRID SIZE (see section 1.5 for definition). The conditions are:

- $\gamma > 90^\circ$ AND $d_{\mathrm{wyp}} <$ GRID SIZE
- $d_{\mathrm{t\_next}} < \mathrm{D}_{\mathrm{maneuver}}$ AND $d_{\mathrm{wyp}} <$ GRID SIZE

$\mathrm{D}_{\mathrm{maneuver}}$ is a predefined distance for allowing maneuvering space, for AVALON typically set to 3 m. These conditions still have to be tested extensively on the boat, since it could end up bad if waypoints are skipped or if the transfer to the next waypoint does not work. The GRID SIZE has yet to be defined, approximately 20 m have given reasonable results while testing on the Urnersee. For the Atlantic Crossing, the GRID SIZE has to be bigger, for short course racing where a high accuracy is needed, a smaller GRID SIZE of about 5 to 10 m would be much better.

New Calculation In order to make sure that the path planner is not constantly running and recalculating, the new calculation will only be triggered in three cases:

- Wind changes more than 20°
- Last calculation is older than 60 minutes
- d_{t} is greater than GRID SIZE

Tests have shown that a wind change of 20° especially on lakes is very likely to happen. However, at the moment we decided to leave the hysteresis on 20° to make sure that the calculated path is accurate and sailable.

3.2.2 New Calculation Mode

This mode mainly controls and supervises the new calculation. In order to do that, it sets a flag (ENABLER) that can be read and interpreted by the planner. By controlling the timestamps of the written waypoints, this state `new calculation` can determine when the planning has finished and then switch back to `standard mode`. During this state, no heading is generated and the controller follows the last heading received.

3.2.3 Destination Approach

If working properly, every path tracking process would end up in this state. The state machine switches from `standard mode` to `destination approach` as soon as the last waypoint is approached. The desired heading that is now transmitted to the controller is the true course from present boat position to target position.

After the execution is finished, the state machine automatically switches back to idle mode.

3.2.4 Buoy Approach

The buoy approach state is a state especially designed for small distance regattas. The rules of the World Robotic Sailing Championship (WRSC 2009) that will be held in Portugal in July require a port side surrounding of all the buoys (see figure

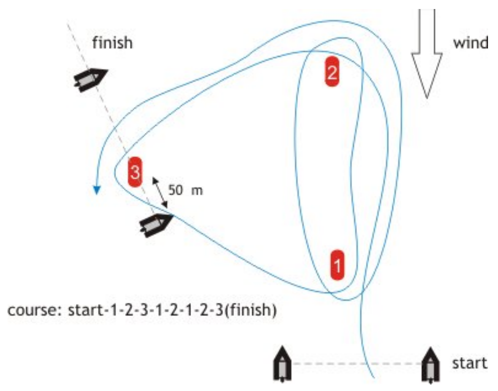


Figure 3.4: Regatta course for WRSC 2009

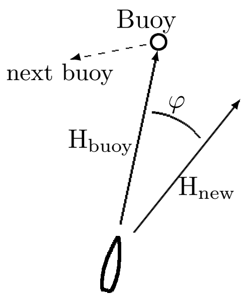


Figure 3.5: Basic buoy approach scheme

3.4). One of the several solutions considered for the path planner to manage that was to set multiple waypoints around the buoy. The problem with that is that if the boat had to tack upwind to get from one small waypoint to the other, AVALON could accidentally pass the buoy on the wrong side. That led to the conclusion that AVALON has to be guided around the buoy. To support the planner, undesired areas can also be mapped as static obstacles.

With that in mind, the state machine automatically switches into the state `buoy approach` when approaching a buoy, which then produces specific headings to sail around the buoy.

Figure 3.5 illustrates the new heading that is generated by adding $\varphi = 30°$ (experimentally defined) to the heading pointing directly towards the buoy.

By checking the flags, the SAILOR also knows that we are approaching a buoy. The controller then tries to follow as precisely as possible the received new heading, but keeps in mind never to perform maneuvers that would result in a new direction to the left of the desired heading.

This state has yet to be tested on board at the lake, simulations will be done extensively in June.

Chapter 4

Trajectory Planner

Although most of the implementation with attention to costs has already been discussed in chapter 2 section V, I would like to highlight some important aspects of the algorithm.

The whole planning cycle is split into a global and a local planning. The global planner's work is to generate a rough mapping from current boat position to the final destination. A detailed path is then produced by the local planner, incorporating only the next few nautical miles along the rough path that was set by the global planner.

4.1 Premises for Path Planning

4.1.1 Heading

We assume a 24 neighbour grid throughout this paper, allowing us to sail in 16 discrete headings. Figure 4.1 shows the 24 nodes that can be reached and all possible directions. Another advantage of being able to expand to the second but next neighbour-grid is the fact that it reduces the number of waypoints, since one grid node can be skipped and smoother paths are generated.

In theory, every heading change is possible with a sailing boat, keeping in mind the space that Avalon needs for turning. However, to make the algorithm faster, we have constricted the biggest possible heading change to 90° (black lines in figure 4.1). The areas that are not sailable due to the wind direction are between $\pm 45^o$ to wind direction (sailing upwind) and $\pm 160^o$ to wind direction (sailing downwind).

4.1.2 Polar diagram

As stated in chapter 2 section E2, we need a polar diagram in order to calculate the boat speed for a specific angle to the wind. Due to not very constant wind conditions during our testing phase at the Urnersee, we were not able to create a specific polar diagram for Avalon. We have therefore created an own polar diagram out of existing diagrams for bigger vessels. Figure 4.2 shows the diagram used for Avalon. Using gnuplot, we set some predefined points and fitted an exponential

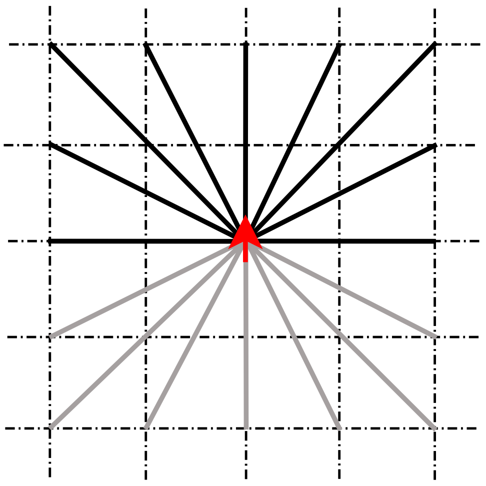


Figure 4.1: Possible 16 different headings that are considered for AVALON. In this case, the grey lines fall away due to the restriction of a maximum heading change of 90°

function on top of it. Equation 4.1 shows the resulting function for speed, α being the respective *angle to the wind* and v_{wind} the current wind speed and v_{boat} the resulting boat speed in knots[1]. Testing has shown that we get reasonable results by restricting the sailable *angle to the wind* to the area between 45° and 160°.

$$v_{\text{boat}} = (-0.0048 \cdot \alpha^4 + 0.0335 \cdot \alpha^3 - 0.2460 \cdot \alpha^2 + 0.7465 \cdot \alpha - 0.01109) \cdot v_{\text{wind}} \quad (4.1)$$

4.1.3 Coordinate Transformation

For the implementation of the planning algorithm in AVALON, the GPS coordinates of current position as well as destination position have to be transformed in a way that the two points could be placed upon a grid. Since we are only planning over short distances (15 to 20 nautical miles on the Atlantic), we can use a very simple approximation (equation 4.2) that assumes a flat water surface [3] and returns meter-coordinates from spherical global coordinates. R_E is the earth radius, lon and lat the GPS coordinates in degrees.

$$\begin{aligned} x &= R_E \cdot \cos(\text{lat}) \cdot \frac{\pi}{180\,deg} \cdot \text{lon} \\ y &= R_E \cdot \frac{\pi}{180\,deg} \cdot \text{lat} \end{aligned} \quad (4.2)$$

Testing has shown that we get fairly reasonable results using this equation. After transforming the coordinates, a certain offset has to be subtracted in order to place either start or end goal close to the origin of the initialized map. To do that, the smaller value of either start or goal was subtracted from both of them, keeping in mind a certain small local offset to give the path some space.

[1] 1 knot is the equivalent of 1 nautical mile per hour

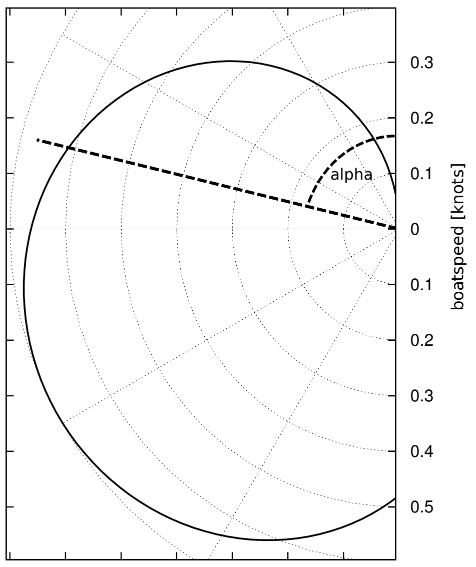


Figure 4.2: Used polar diagram for 1 knot of wind speed

4.1.4 θ at Start and Target Node

An interesting topic as well is the automatic generation of a heading (θ) for the target node. In order to start the search, A* needs to know the full goal state, including θ at the end. Either this is created by a logic or the algorithm is adapted to stop the search if x and y of the goal state is reached. We decided on the former approach and developed a logic that generates a target θ that depends on the wind bearing and the relative position of the start node.

Initially, the θ is defined as the direct bearing from start to target node. If the difference to the wind bearing is greater than 45°, this θ is taken, otherwise it is shifted until the condition is met.

The only problem with that approach is that obstacles are not considered. If the path is forced to approach the target from a specific side, it is possible that the θ differs 180° from the approaching angle and therefore requests another maneuver (compare with chapter 2, section V.H.)

On the Atlantic, this does not cause problems since a replanning is always triggered before reaching the destination. For short course racing the θ just has to be set manually to a reasonable angle.

4.2 Global Planning

The initial notion was to implement large scale weather forecasts in order to choose a global route that takes advantage of good weather situations and avoids stormy regions. Wind data from a sophisticated weather model by MeteoNews could have been sent in form of bitmap pictures over email and our satellite communication system to Avalon. Unfortunately, analysis have shown that the benefits of using this information for our purposes are so small that it is not worth implementing it. Considering an average boat speed of about 3 to 4 knots, we would never be able to avoid a storm that is moving twenty times as fast as AVALON.

To conclude, we have decided not to use weather forecasts for our global planning. Instead, we are going to set fixed global waypoints in advance, based on current analysis, average weather information and expert knowledge. An artificial bandwidth will be defined to make sure that the boat will not approach islands or dangerous areas (i.e. Bay of Biscay) and stay within the preset route.

4.3 Local Planning

4.3.1 Planning Algorithm

As explained earlier (chapter 2 section V.E), the local planner is based on a very well established and robust A* algorithm. In principle, A* expands from one starting node to its neighbours, allocating costs for moving to that specific neighbour. The algorithm then expands from the neighbour with the smallest cost value to all of his neighbours. In the end, the optimal path can be extracted by following the nodes with the smallest cost values from target to start.

4.3.2 Cost Settings

In order to get paths that are optimized for speed (minimal execution duration), the unit chosen for the allocated costs is *traveling time*. This section lists all the different costs and sub functions used.

General Cost

The general aim of this algorithm is to find the fastest path for a sailing vessel to go from point A to point B. The general cost we give for every movement is therefore *sailing time to the next node*, calculated by equation 4.3. The speed depends on the angle to the wind and the wind speed and is calculated with our polar diagram (see section 4.1.2).

$$\text{sailing time} = \frac{\text{distance to next node}}{\text{speed}} \tag{4.3}$$

Turning Cost

To generate very smooth paths, the algorithm places costs C_{turn} for heading changes, using a simple linear factor f in equation

$$C_{\text{new}} = f \cdot |\theta_{\text{curr}} - \theta_{\text{next}}| \tag{4.4}$$

where θ_{curr} and θ_{next} are the respective headings. An iterative process to determine the number of f is described in section 5.3.

Cost for Tack and Jibe

To be able to control more subtly the number of maneuvers that have to be done in a path, we added additional costs for tack and jibe. With Φ being the wind angle and θ the respective headings, we can determine with the inequation

$$(\theta_{\text{curr}} - \Phi) \cdot (\theta_{\text{next}} - \Phi) < 0 \tag{4.5}$$

whether a maneuver would have to be undertaken. If the condition is true, either tack or jibe have to be made in order to adjust to the new heading and a certain cost will be added.

Heuristic Estimate

The heuristic estimate is used to evaluate the remaining time to the target node. According to [2], the only constraint for the heuristic in order to still get optimal results is that it has to underestimate the cost of the remaining path. To match the units of our general costs (time), the heuristic has to be a duration estimate for the remaining path. The estimate used in our implementation is based on a very conservative assumption of an average sailing speed of 7 knots over the remaining distance (straight line) without any consideration of current wind direction and speed. Like this we make sure that the estimated remaining time is always smaller than the optimal real sailing time.

Tunnel cost

To favour paths that stay close to the straight connection of start and goal node, we allocate costs increasing with the distance to that direct trajectory. If inserted, a logic has to be defined that sets a reasonable tunnel width. Setting a predefined fixed tunnel width is not sufficient since an appropriate width depends significantly on the overall path length. We have tested several different approaches for the increasing tunnel cost:

- Linear increased cost
- Cost increase along a preamble
- No increase inside the tunnel but significant costs at the boundaries (tunnel)

The linear increase has proved best of the three methods and was used for the plot in fig 5.3b by using the following equation 4.6, C being the added costs, $d_{\text{trajectory}}$ the shortest distance to the straight connection of start and goal:

$$C = \frac{d_{\text{trajectory}} \cdot \text{GRID SIZE}}{\text{TUNNEL WIDTH}} \cdot 10 \tag{4.6}$$

4.3.3 Obstacle Avoidance

Obstacle avoidance is a topic that has already been discussed in many papers [2], especially using visual aids like cameras [4]. It is becoming a more and more interesting research topic, especially when it comes to interactions between several autonomous systems. Part of the goal for this work is also to implement a collision avoidance system and to be able to plan reasonable paths around static and dynamic obstacles.

Static Obstacles

Static obstacles like islands and coastal areas can be determined and implemented in advance. By fixing one GPS coordinate on a map and then parsing it into our embedded system, static obstacles can be handled easily and with a certain security margin.

Dynamic Obstacles

Dynamic obstacles on the other hand are much harder to implement. The following lines will give an overview of the information Avalon obtains, general problems with dynamic obstacles and how they are handled on board.

AIS Data Equipped with a VHF Antenna that can receive AIS Information, Avalon is able to receive signals with detailed status reports of ships being in VHF range of our boat. Processing this information, we are able to know their exact position in GPS coordinates, their speed in knots and their current heading.

Main Problem The biggest issue for us with dynamic obstacles is that our planning is not time-specific enough to make estimations about our location in a specific future time. This means that moving ships or obstacles have to be viewed as static obstacles, enlarged with a certain specific factor for safety reasons. The important part is to find a feasible representation for all the boats in order to get more or less constant path results. In detail, this means that the generated path needs to look the same at calculation time $t^{(0)}$ (information about the moving ship is already received, the ship is still far away) and $t^{(1)}$ (ship is much closer now, maybe right in front of us). That would mean that the static obstacle (for that specific ship) has to look more or less the same and be at the same position at both times $t^{(0)}$ and $t^{(1)}$. This is not a necessity but it would be much nicer since the produced paths would always look more or less the same.

Possible Solutions There are several solutions that could be taken into account:

1. We could shorten the time intervals between replanning significantly, using a very small square obstacle for ships' current location. This way we can picture the moving ships with small static obstacles, which would affect the resulting path less than big obstacles.

2. The path planner could calculate paths that are not affected by any information about moving ships. The skipper (chapter 3) would then act as a watchdog that controls the movements of surrounding ships. Since the skipper knows the path for Avalon, it could for example calculate the expected crossing time of surrounding ships for our path and evaluate, if a recalculation with a different temporary target node is necessary.

3. In order to represent the ships as small as possible, we need to estimate in advance the position where a possible crash could happen. To do that, we could step by step check specific positions on the ship's future trajectory. The trajectory can be calculated with the known AIS information about speed and heading. The condition 4.7 only turns true for possible points of impact. A static obstacle could then be positioned at that point. Numerator $d_{\text{ship}}^{(t)}$ represents the distance between Avalon and the projected ship at time t.

$$t - \frac{d_{\text{ship}}^{(t)}}{\bar{v}_{\text{Avalon}}} \quad < \quad 20 \quad [\text{sec}] \tag{4.7}$$

4. Another possibility could be that the skipper does not trigger a replanning when sensing danger but tries to maneuver Avalon around the obstacles when confronted. This would be a similar approach to the buoy-problem discussed in section 3.2.4

Since the basics for an implementation of the latter solution have been laid already with the regatta-configuration (`buoy approach mode`), the solution where the skipper tries to navigate around (moving) obstacles to avoid imminent crashes seems reasonable and feasible. The implementation into the navigation system of Avalon has not been done yet due to time constraints. Testing will be done in a first case by implementing artificial dynamic obstacles in our self developed simulator, later by replaying logged or simulated AIS data on the actual boat.

Chapter 5

Tests and Analysis

This section quickly discusses the used testing environment and then focusses on the optimization of various parameter sets for a defined setup with obstacles and different wind conditions.

5.1 Testing Environment

To test the whole software, several steps were attempted. In a first round, we wanted to get good path results without looking on the performance of the algorithm. This was done by visualizing the generated paths with gnuplot. In a second round, the algorithm was integrated into the DDX environment where teething problems were solved with a self written DDX simulator that produces artificial sensor data. With success we then tested the whole planning algorithm on the boat computer MPC21. During testing days at the Urnersee, we were able to test the whole system for the first time with all the environmental sensors and systems running, setting a desired destination point and letting the boat plan and execute a path to that destination. Further tests are planned to take place on Lake Constance, the Lake of Zurich and the Atlantic Ocean.

5.2 Testing Lists

Before each test, a list is created with things that have to be tested. Every element has to meet specific criteria in order to continue. Following a list with items that have not been tested yet successfully and will be part of the test of June 2009:

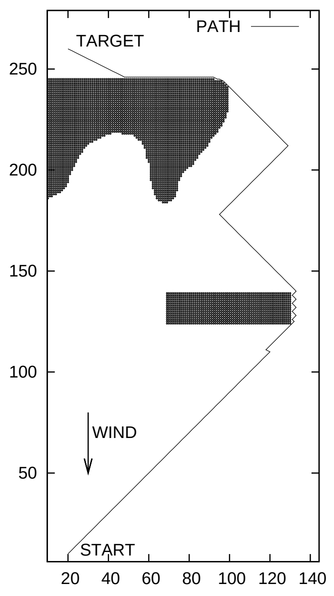
- Reach a certain destination upwind / downwind

  Successful if distance to target position is less than GRID SIZE after the skipper declares to be finished.

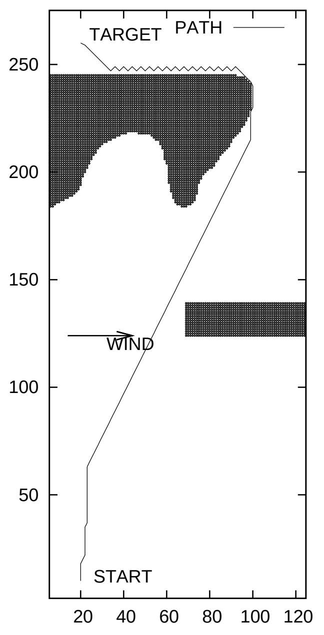
- Buoy Maneuver

  Successful if AVALON can sail the same course (see figure 3.4) for 5 times without missing one buoy.

- Long time navigation



(a)



(b)

Figure 5.1: Path solutions only with general costs

Successful if AVALON can sail without human interaction for at least 24 hours.

- Obstacle avoidance

  Successful if 4 dynamic obstacles (simulated ships) can be considered at once without recalculating every 20 seconds. The resulting paths of every recalculation will be evaluated and have to look more or less the same.

5.3 Parameter Optimization

Figure 5.1a and 5.1b show path results with implemented general costs. This means that the heading is always sailable in respect to the wind but no costs have yet been added for heading changes and so on. The tunnelcost and the heuristic estimate are set to zero as well. With the chosen setup of two obstacles, the effects of adding heuristic and tunnelcost can be presented very well. The square obstacle could for example represent a ship on the ocean (section 4.3.3 for more details on obstacle representation), the big round obstacle an island.

Figure 5.1a and 5.1b show well that we need a turning cost in order to generate much smoother paths with much fewer turns. As section 4.3.2 already explains, adding a turning cost was achieved by implementing equation 4.4. The value 8 for the loading factor f was found to yield good results. Although wind speed is only a factor for general costs and does not affect the turning cost, it is interesting to see that the wind speed has no influence on the path in a way that the result would get worse.

The result of adding a cost for heading changes is shown in figure 5.2 for the exact same setup as before. Experiments have been made with adding costs for tack and

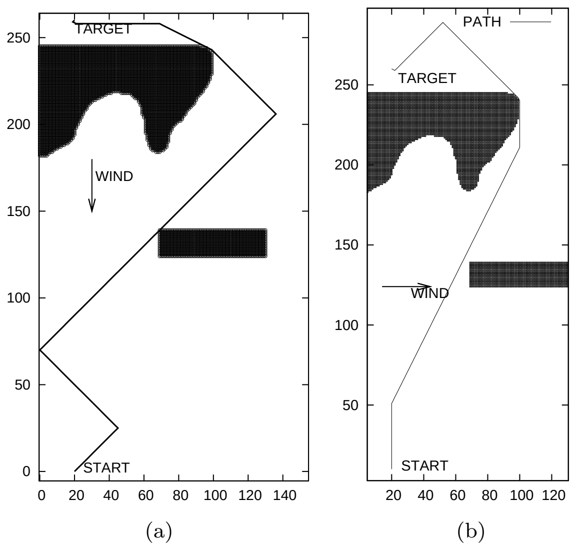


Figure 5.2: Paths with added cost for heading changes

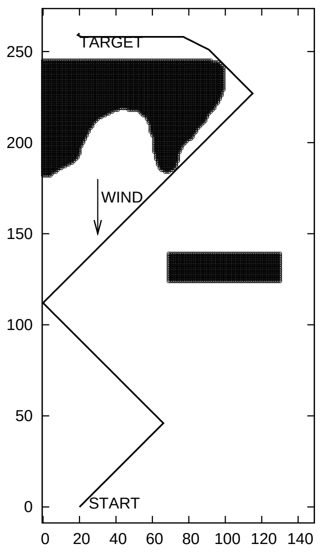
jibe (see section 4.3.2) but it was found that it is not necessary because it did not add the expected benefit.

Heuristic Estimate The next step is to add the cost for the heuristic estimate as well as the tunnel cost and evaluate them. Figure 5.3a shows the addition of a heuristic estimate, using a boat speed of 7 knots to sail the remaining path (see section 4.3.2 for details). It can be seen very well that the path is in an early state much closer to the target already but does not get worse by that. The number of tacks stays exactly the same and the path would be very well sailable. The calculation time needed with additional heuristic (A* configuration) is about 10 to 25% less.

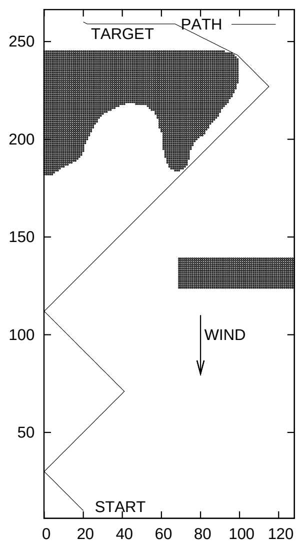
Tunnel Cost A tunnel cost that is raised for increasing distance to a straight connection between start and goal node affects the resulting path in a negative way (see figure 5.3b). The biggest problem with adding tunnel costs arises when obstacles are in the way like on the used setup. To smooth the path again and reduce the number of tacks, costs for heading changes have to be elevated.
The tests have shown that adding additional costs like tunnel costs or costs for tack / jibe do not enhance the path solutions at all. Since the maximal width of the path over long distances (Atlantic crossing) is defined by the bandwidth of the global planning (section 4.2), we have decided to use it only in short course races where a tactical decision on the tunnel width can be made in advance.

The heuristic estimate, which is basically the difference between Dijkstra and A*, directs the path earlier towards the goal which makes the algorithm run faster. Especially on Avalon's board computer, this is extremely advantageous.

A new setup tested with several wind directions is shown in figure 5.4. 5.4b shows for the first time limitations of using only 16 different headings. After passing the obstacle, it would be better to head directly for the target node. The first part of the path in figure 5.4c highlights well the restrictions of a maximal *angle to the*



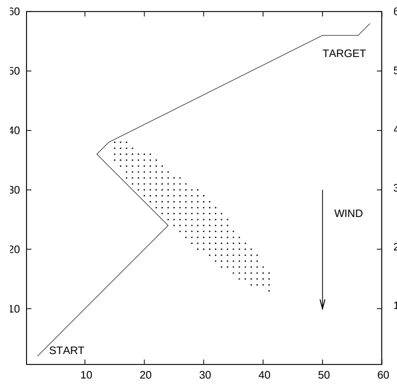
(a) Added heuristic



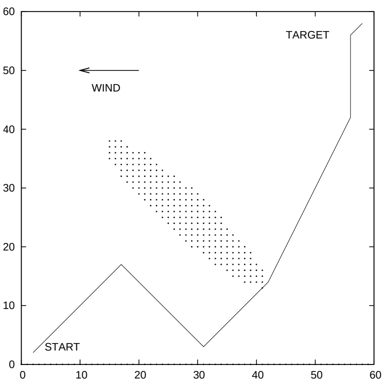
(b) Added tunnel cost

Figure 5.3: Additional costs on the setup of fig 5.2a

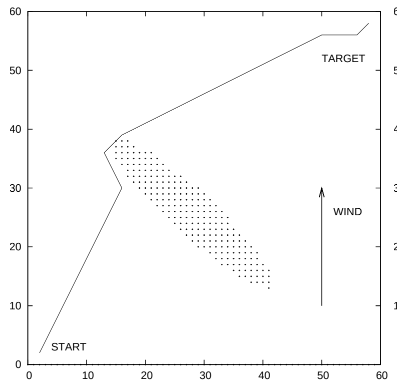
wind of 160° degrees for downwind sailing.



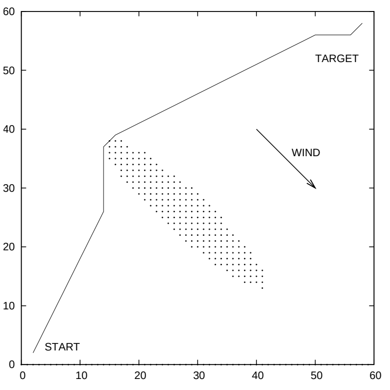
(a)



(b)



(c)



(d)

Figure 5.4: Path solutions for different wind directions

Chapter 6

Conclusion and Outlook

6.1 Conclusion

To conclude, it can be said that the aims of this bachelor thesis have been accomplished in a way that an autonomous crossing of the Atlantic Ocean can safely be approached. The planner is able to generate smooth and sailable paths as well as avoid static obstacles like islands and coastal regions that can be mapped in advance.

The execution of the planning on the existing hardware has been successfully tested with runtime tests and debugging tools. This leads to the results that the planner does not use too much computing power and time in order for other programs to continue their work. Some problems have been encountered with the memory size when initializing a grid. In order to remove this weak spot, conditions have to be inserted that only allow a certain grid size to be initialized. Although simulations show very good results, the software still has to be tested extensively on the water. For us it is important to generate artificial environmental conditions (faked storm, wrong IMU-Data . . .) in order to get to know the weak spots of the software.

Concerning the software engineering, we have seen that the initial notion to use an A* algorithm has proved well and finally produces sailable and reasonable results. Adding tunnel costs to narrow the width of the resulting path was seen to be rather obstructive and not supporting in terms of path optimality. The maximal width of the generated path on the Atlantic will be controlled by the predefined "allowance band" of the global planning. For short course racing, we will define a certain tunnel width to make sure that the boat is not tacking upwind too far away from the direct start - target connection.

6.2 Outlook

During this summer 2009 and especially before the start of the Microtransat, several things have yet to be tested, enhanced and optimized. First of all, remaining code parts like dynamic obstacle avoidance as well as short course racing parameters have to be implemented and tested. Competing at the World Robotic Sailing Championship (WRSC 2009) in July will give us the opportunity to exchange ideas with our competitors, but also - being in Portugal - enable us to test Avalon for the first time at sea.

Another important part on the testing agenda will be endurance tests. Since the system has to be up and running for approximately three months during the transatlantic crossing, all the programs have to be checked for long time runtime errors. These tests will mainly be done on the Lake Constance where the infrastructure for endurance tests exists.

Bibliography

[1] Erckens, Hendrik: Design of a Trajectory Following Controller for an Autonomous Sailboat / ETH Zürich. June 2009. – Forschungsbericht. Bachelor Thesis

[2] LaValle, S.M.: *Planning algorithms*. Cambridge University Press, 2006

[3] Stelzer, R. ; Pröll, T.: Autonomous sailboat navigation for short course racing. In: *Robotics and Autonomous Systems* 56 (2008), Nr. 7, S. 604–614

[4] Wu, C.P. ; Lee, T.T. ; Tsai, C.R.: Obstacle avoidance motion planning for mobile robots in a dynamic environment with moving obstacles. In: *Robotica* 15 (1997), Nr. 05, S. 493–510

List of Figures