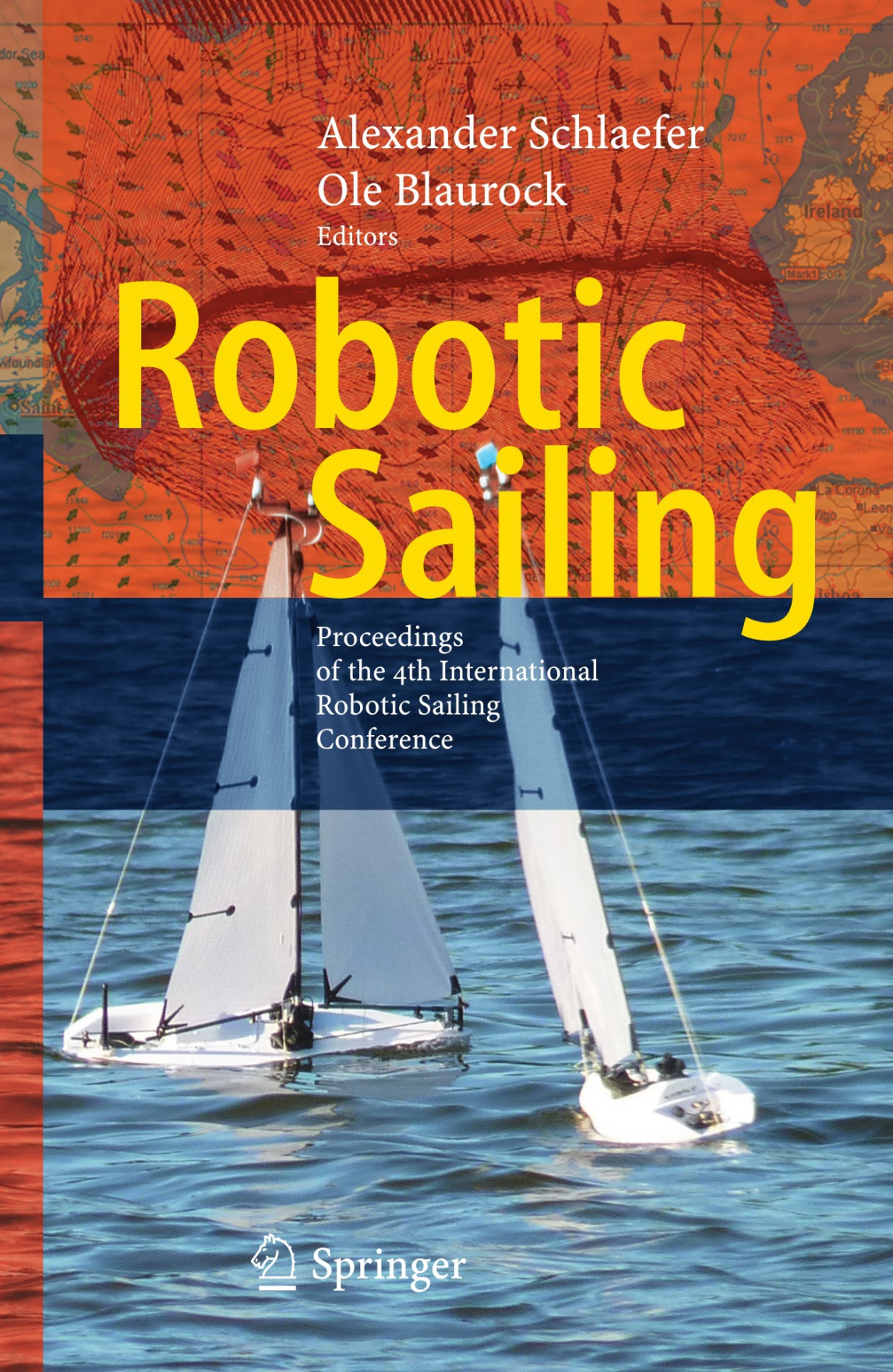

Alexander Schlaefer
Ole Blaurock
Editors
Robotic Sailing
Proceedings
of the 4th International
Robotic Sailing
Conference
Springer

Robotic Sailing

Alexander Schlaefer and Ole Blaurock (Eds.)

Robotic Sailing

Proceedings of the 4th International Robotic Sailing Conference

Springer

Editors

Prof. Dr. Alexander Schlaefer
University of Luebeck
Institute for Robotics and
Cognitive Systems
Ratzeburger Allee 160
23538 Lübeck
Germany
E-mail: schlaefer@rob.uni-luebeck.de

Prof. Dr. Ole Blaurock
University of Applied Sciences Luebeck
Fachbereich Elektrotechnik
und Informatik
Mönkhofer Weg 239
23562 Lübeck
E-mail: blaurock@fh-luebeck.de

ISBN 978-3-642-22835-3 e-ISBN 978-3-642-22836-0

DOI 10.1007/978-3-642-22836-0

Library of Congress Control Number: 2011933572

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

IRSC 2011 is only the fourth in a series of conferences dedicated to robotic sailing, and while still comparatively small, we have seen a substantial increase in the number of groups interested in and working on robotic sailboats recently. Given that sailing is a fairly old way of locomotion and a high-tech sports today, it is somewhat surprising that the first competition for autonomous sailboats was proposed as late as 2004. Yet, the original objective to autonomously sail across the Atlantic ocean proved to be fairly ambitious, and no boat has succeeded so far.

However, this also highlights the complexity of the engineering challenges at hand. Sailing depends as much on the physical properties of boat and rig as on the course and route set in the context of changing winds and currents. Moreover, performance measures do not only include the boat speed, but also seaworthiness and robustness of the whole system. Hence, building a robotic sailboat is a true interdisciplinary project, involving naval architecture and physics, electrical engineering and power managment, embedded systems, computer science, and systems engineering.

Establishing a conference held jointly with the *World Robotic Sailing Championship* (WRSC) has provided a platform for discussions among scientists from all fields involved in robotic sailing. In fact, we believe that the progress made in autonomous sailing so far is to no small extent driven by this combination of competition and knowledge exchange. The interdisciplinary nature of robotics and robotic sailing is reflected in the papers contributed to IRSC and the teams participating in WRSC. Further promoting this multidisciplinary approach will be key to tackling the numerous challenges on the way to truely autonomous sailboats.

These proceedings summarize the state of the art in robotic sailing, and the introduction in Part I contains a review illustrating its history and recent advances. Clearly, having a robust and reliable boat is a key requirement, which is also the focus of papers in Part II. The proposed designs range from small one-design boats for algorithm development to vessels built to cross the Atlantic Ocean. Different aspects of the system design and validation are discussed in Part III. The remaining papers focus on algorithmic matters: Part IV presents approaches for collision avoidance while Part V addresses localization and route planning.

Organizing IRSC 2011 was only possible with the help of many people. We are grateful to all of them, particularly to Petra Roßkopf for her assistance with the conference location, and to our student co-organizers, who have worked tirelessly to get everything arranged. We also thank our sponsors and partners without whom this conference would have been infeasible. Alexander Schlaefer is very appreciative for the patience and tolerance of Achim Schweikard, who supported the idea to organize IRSC/WRSC in Lübeck. Ole Blaurock would like to express his appreciation to the robotic sailing team at the Fachhochschule Lübeck, which started only in the beginning of the year and yet managed to activate numerous colleagues, enabling the participation in WRSC 2011.

Lübeck,
August 2011

Alexander Schlaefer
Ole Blaurock

Organization

General Chair

Alexander Schlaefer	University of Lübeck, Germany

Co-Chair

Uwe Krohn	University of Applied Sciences, Lübeck, Germany

Proceedings

Alexander Schlaefer	University of Lübeck, Germany
Ole Blaurock	University of Applied Sciences, Lübeck, Germany

Secretariat

Cornelia Rieckhoff	University of Lübeck, Germany

Student Co-Organizers

Niko Ammann	University of Lübeck, Germany
Lenka Hanesová	University of Applied Sciences, Lübeck, Germany
Florian Hartmann	University of Lübeck, Germany
Philipp Jauer	University of Lübeck, Germany
Julia Krüger	University of Lübeck, Germany
Tobias Meyer	University of Lübeck, Germany

Program Committee

José C. Alves	Universidade do Porto, Portugal
Ralf Bruder	University of Lübeck, Germany
Nuno A. Cruz	Universidade do Porto, Portugal
Erik Maehle	University of Lübeck, Germany
Benedita Malheiro	ISEP - IPP, Portugal
Paul Miller	USNA, United States of America
Mark J. Neal	University of Aberystwyth, United Kingdom
Cedric Pradalier	ETH Zürich, Switzerland
Colin Sauze	University of Aberystwyth, United Kingdom
Roland Stelzer	INNOC, Austria

Contents

Part I: Introduction

History and Recent Developments in Robotic Sailing 3
Roland Stelzer, Karim Jafarmadar

Part II: Robotic Sailboats

Sailing without Wind Sensor and Other Hardware and Software Innovations 27
Jan Sliwka, Jeremy Nicola, Remi Coquelin, Francois Becket de Megille, Benoit Clement, Luc Jaulin

MOOP: A Miniature Sailing Robot Platform 39
Colin Sauzé, Mark Neal

Breizh Spirit, a Reliable Boat for Crossing the Atlantic Ocean 55
Richard Leloup, Frédéric Le Pivert, Sébastien Thomas, Gabriel Bouvart, Nicolas Douale, Henry De Malet, Laurent Vienney, Yvon Gallou, Kostia Roncin

A New Class for Robotic Sailing: The Robotic Racing Micro Magic 71
Alexander Schlaefer, Daniel Beckmann, Maximilian Heinig, Ralf Bruder

Part III: System Development

A Systems Engineering Approach to the Development of an Autonomous Sailing Vessel 87
Bradley E. Bishop, Joseph Bradshaw, Cody Keef, Nicholas Taschner

Using ARM7 and µC/OS-II to Control an Autonomous Sailboat 101
Michael Koch, Wilhelm Petersen

Simulating Sailing Robots 113
Colin Sauzé, Mark Neal

Part IV: Collision Avoidance

Automatic Obstacle Detection for USV's Navigation Using Vision Sensors 127
Oren Gal

Rule-Compliant Navigation with Qualitative Spatial Reasoning 141
Diedrich Wolter, Frank Dylla, Arne Kreutzmann

Global Data Storage for Collision Avoidance in Robotic Sailboat Racing – The World Server Approach 157
Nikolaus Ammann, Florian Hartmann, Philipp Jauer, Julia Krüger, Tobias Meyer, Ralf Bruder, Alexander Schlaefer

Part V: Localization and Route Planning

A Digital Interface for Imagery and Control of a Navico/Lowrance Broadband Radar 169
Adrian Dabrowski, Sebastian Busch, Roland Stelzer

Route Planning for a Micro-transat Voyage 183
Peter Gibbons-Neff, Paul Miller

A Rule-Based Approach to Long-Term Routing for Autonomous Sailboats 195
Johannes Langbein, Roland Stelzer, Thom Frühwirth

Author Index 205

Part I
Introduction

History and Recent Developments in Robotic Sailing

Roland Stelzer and Karim Jafarmadar

Abstract. Robotic sailing boats represent a rapidly emerging technology for various tasks on lakes and oceans. In this paper we give an overview about the main building blocks of a robotic sailing boat for controlling the rudder and the sails. History of robotic sailing includes developments in mechanical, electronic, and intelligent self-steering systems as well as automatic sail control. Furthermore advantages and disadvantages of rigid wing sails in comparison to traditional fabric sails are illuminated. Early examples of robotic sailing boats and recent developments, stimulated by robotic sailing competitions such as Microtransat Challenge, SailBot and World Robotic Sailing Championships are presented. We conclude with a brief outlook on potential applications in the field of robotic sailing.

1 Introduction

Autonomous sailing robots perform the complex tasks of sailing boat navigation fully automatically and without human assistance. Bowditch [13] defines navigation as *"the process of monitoring and controlling the movement of a craft or vehicle from one place to another"*.

Robotic sailing boats therefore have to perform the complex planning and manoeuvres of sailing fully automatically and without human assistance. Starting off

Roland Stelzer
INNOC – Austrian Society for Innovative Computer Sciences,
Haussteinstraße 4/2, 1020 Vienna, Austria
e-mail: roland.stelzer@innoc.at
Centre for Computational Intelligence, De Montfort University,
Leicester, LE1 9BH, United Kingdom
e-mail: rstelzer@dmu.ac.uk

Karim Jafarmadar
INNOC – Austrian Society for Innovative Computer Sciences,
Haussteinstraße 4/2, 1020 Vienna, Austria
e-mail: karim.jafarmadar@innoc.at

by calculating an optimum route based on weather data and going on to independent tacking and jibing and avoiding collisions, stand-alone sailing boats are able to sail safely and reliably through to any and every destination. The human being merely has to enter the destination co-ordinates.

The key characteristics of a robotic sailing boat can be summarized as follows:

- Wind is the only source of propulsion.
- It is not remotely controlled; the entire control system is on board.
- It is completely energy self-sufficient; this is not a must in the sense of definition of a robotic sailing boat, but it opens a wide range of applications.

Although many technical aids are available for common sailing boats, relatively little effort has been spent on research of autonomous sailing. Research on autonomous surface vehicles (ASV) was mainly focused on short-range crafts powered by electrical or combustion engines. Such crafts are limited in range and endurance depending on the amount of fuel or battery capacity on board to run a motor for propulsion. In contrast a sailing vessel needs only a minimal amount of power to run sensors, computers and to adjust sail and rudder position.

Extensive research has been undertaken on semi-autonomous systems, where just a subset of the functionality of a robotic sailing boat is covered. The history of self-steering gears and automatic sail control will be discussed independently in the following sections. Afterwards a separate section shows history and recent research projects in completely autonomous sailing.

2 History of Robotic Sailing

2.1 Self-steering Gear

Historically, the first task to be automated was the governing of the rudder. A self-steering gear is an equipment used on ships and boats to maintain a chosen course without constant human action. Self-steering gear is also referred to as *autopilot* or *autohelm*[1]. Basically the different forms of self-steering gears can be divided into two categories: mechanical and electronic.

2.1.1 Mechanical Self-Steering

Fishermen who bind the rudder or tiller of their boat in a fixed position to produce an optimal course can be seen as a first approach to a mechanical self-steering system [37].

A more sophisticated mechanical approach is the *wind vane* developed first by Herbert Hasler (1914-1987), who is known as one of the fathers of single-handed sailing[2]. Wind vanes are now sold by a number of manufacturers, but most share

[1] Autohelm is a Raymarine trademark, but often used generically.

[2] Single-handed sailing is sailing with only one crew member. The term is usually used with reference to ocean and long-distance sailing.

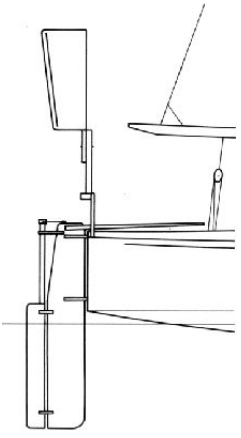


Fig. 1 Example for a wind vane with trim tab on main rudder (from [41])

the same principle: The device consists of a wind vane secured at the stern of the yacht, which is connected to the rudder respectively a trim tab on the rudder via a system of ropes, pulleys and servos (see figure 1). When the angle of the apparent wind[3] changes, this change is registered by the air vane, which activates the steering device to return the boat to the selected point of sail[4]. Wind vane self-steering does not steer a constant compass course but a constant point of sail.

2.1.2 Electronic Self-steering

Electronic self-steering controls the rudder movement by electronics based on various sensor input values. At least a compass is necessary; additional sensors can deliver wind direction or GPS position in order to calculate a heading towards a given target waypoint.

Substantial progress toward automatic steering was based on the invention of electronic gyrocompasses. The earliest known gyroscope-like instrument was made by German Johann Bohnenberger, who first wrote about it in 1817 [12]. According to Bennett [9] and Roberts [37] the major contributions to the development of a practical automatic steering system were made by Sperry Gyroscope Company. Elmer Sperry developed his first automatic ship steering mechanism in 1911 [3, 46]. Sperry's gyropilot was known as *Metal Mick* as it was capturing much of the behaviour of an experienced helmsman. It compensated for varying sea states using feedback control and automatic gain adjustments. This lead to a first simple adaptive autopilot.

The work of Minorsky [32] is also regarded as making key contributions to automatic ship steering. Nicholas Minorsky presented a detailed analysis of a position

[3] Apparent wind is referred to as the velocity of air as measured from a moving object, as a ship.

[4] Point of sail describes the direction of a boat with respect to the direction of the wind.

feedback control. He formulated the specification of a three-term controller, better known as proportional-integral-derivative (PID) controller.

2.1.3 Intelligent Rudder Control

Conventional electronic self-steering systems found on the majority of vessels at sea still employ PID control algorithms to control the heading [18]. Van Amerongen [52] identified two major disadvantages of this type of controller:

1. It is difficult to adjust manually, because the operator usually lacks the insight into control theory.
2. The optimal adjustment varies and is not known by the user. Changing circumstances require manual readjustment of a series of settings.

Due to the highly dynamic and ever-changing environment, artificial intelligence (AI) techniques, like fuzzy logic (FL), artificial neural networks (ANN), and combinations thereof have received considerable attention for rudder control on ships. Various publications have shown the suitability of FL for rudder control [55, 1, 56, 49]. Polkinghorne et al. [36] furthermore made a comparison to its conventional PID controlled equivalent. The experiments have shown a much smoother rudder action for the FL controller. Stelzer et al. [49] again demonstrated a reasonable performance of a FL controlled rudder, even during tacking and jibing.

Adaptive FL controllers have been presented as a promising approach to combine expert's knowledge and new experiences automatically. In Velagic et al. [53] a Sugeno type fuzzy inference system is combined with a feedback loop to adjust scaling factors of the base fuzzy system. For the mentioned FL approaches human expert's knowledge must be known a priori to design the fuzzy rule set. In contrast, Layne and Passino [26] published a learning control algorithm which automatically generates the fuzzy controller's knowledge base on-line as new information on how to control the ship is gathered. Other examples of adaptive rudder control systems are based on artificial neural networks. Both Enab [24] and Burns [18] aim to provide an ANN-based system that can adapt its parameters towards optimal performance over a range of conditions without the need of manual adjustments.

An ambitious machine-learning approach to automatic rudder control was the *RoboSail* project [51]. The project started in 1997 with the aim to build a self-learning autopilot for a single-handed sailing yacht. Agent technology, machine learning, data mining, and rule-based reasoning have been combined to a system which became commercially available after five years of development [2].

A rudderless approach on automatic heading control of a sailing boat was presented by Benatar et al. [8]. They have shown, that control of a rudderless boat with two sails can be achieved by coordinating the two sails for propulsion and turning.

2.2 Automatic Sail Control

While extensive research has been carried out on automatic steering devices, automatic sail trim is a more recent idea and not yet well covered by scientific publications.

2.2.1 Rigging and Sails

So far several different riggings[5] have been used on robotic sailing boats. They can be characterised according the following criteria:

- **Material and shape:** traditional fabric sail or rigid wing sail
- **Rigging:** Balanced or unbalanced

In history of sailing, which lasts several thousands of years, a large variety of different sail shapes and technologies have been used. Virtually all boats apart from the recent sailing history used conventional fabric sails. This form of sails have some advantageous properties, especially when controlled by a human sailor. This includes the easy way of reefing, repairing, or that shape and camber can be altered by simply tensioning and releasing control lines.

I contrast, a wing sail is a rigid surface with an aerofoil cross-section similar to an aircraft wing. It can provide a much better lift-to-drag ratio than conventional sails [43]. Neal et al. [35] state as significant disadvantages of a wing sail that it is extremely difficult to design it in a way that it can be reefed reliably. Furthermore to construct strong, lightweight rotatable wings at reasonable costs is mentioned there to be difficult. However, they maintain after extensive testing with different wing sails that the potential gains in reliability and efficiency outweigh these problems.

Although most of autonomous sailing boats featuring wing sails have been either designed for longevity [35] or precision sailing [22] rather than for performance, the America's Cup 2010 has shown impressively the dynamic abilities of a rigid wing sail. The trimaran USA-17 (formerly known as BMW Oracle Racing 90 or BOR90) won the trophy with a rigid wing as its main sail.

On a conventional sloop rig, which is the most common rig type on sailing vessels, relatively high power is needed to tighten the sails against wind force. As being self-sufficient in terms of energy is one of the major goals in robotic sailing, the rig design has been put into the focus of attention. A balanced rig design (also known as Balestron rig, Aerorig^{TM}, swing rig, and EasyRig^{TM}) provides great potential to save power [33, 7]. A balanced rig consists of an unstayed mast carrying a main and jib (see Fig. 2). The main boom extends forward of the mast (the mast passes through the boom) to the tack of the jib. The main and jib are sized so that the force from the mainsail is slightly higher than that from the jib. That is, the combined center of effort is just behind the mast. Therefore the load on the sheets is reduced by more than 50 % compared to a conventional rig due to the balanced distribution of the sail load caused by wind [25].

Balanced rigs have been used on the autonomous sailing boats *Avalon* [25] and *IBoat* [14]. Furthermore, most of the rigid wing sails mentioned above can be considered to be balanced rigs.

[5] Rigging is the mechanical sailing apparatus attached to the hull in order to move the boat as a whole. This includes cordage, sails, and spars (masts and other solid objects sails are attached to).

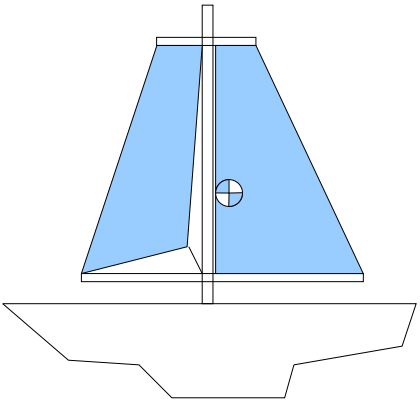


Fig. 2 Balanced rig example: the combined center of effort is just behind the mast

2.2.2 Sail Control Strategies

Most sail control strategies published for autonomous sailing boats rely on locally measured apparent wind data only [1, 17, 25]. While many of them have a virtually infinite number of sail positions (limited just by the resolution of the actuator or the used data types) and therefore allow smooth sail control, just 10 discrete sail position are used on *MOOP* (University of Aberystwyth, UK) featuring a hysteresis condition to avoid permanent switching between two adjacent positions [17]. Reasons for a reduced number of sail positions are to save power on the sail actuator and to extend the lifetime of the sail gear. A state machine to allow for special sail trim during manoeuvres such as tack and jibe has been implemented on *Däumling* (University of Lübeck, Germany) *Avalon* (Swiss Federal Institute of Technology Zurich, Switzerland) and *IBoat* (ISAE, France) [25, 14, 17].

A method which does not directly calculate a sail position based on wind data was published by Stelzer et al. [49]. It firstly determines a desired heel[6] for the boat out of speed and direction of the apparent wind. A feedback-loop implemented as a Mamdani type fuzzy inference system then controls the sail position towards this heel value.

All methods described above can basically be applied to both conventional and wing sails. For the latter a further control method has been published, namely a self-trimming wing sail. A self-trimming arrangement typically consists of a wing sail vertically mounted in bearings that allow free rotation. A smaller wing called *tail* is usually mounted just behind the main wing (see figure 3). An aircraft uses tails to control the exact angle of attack of its wings. Similarly, the tail on a wing sail system is able to control the thrust obtained from the wind and will automatically take into account any changes in wind direction [54]. Extensive research on self-trimming wing sails have been carried out by Elkaim and Boyce [23]. Their experiments have shown upwind progress at $20-25$ deg and speeds of 60 % of the

[6] Heeling is the sidewards tilt of a sailing boat usually caused by lateral wind force.

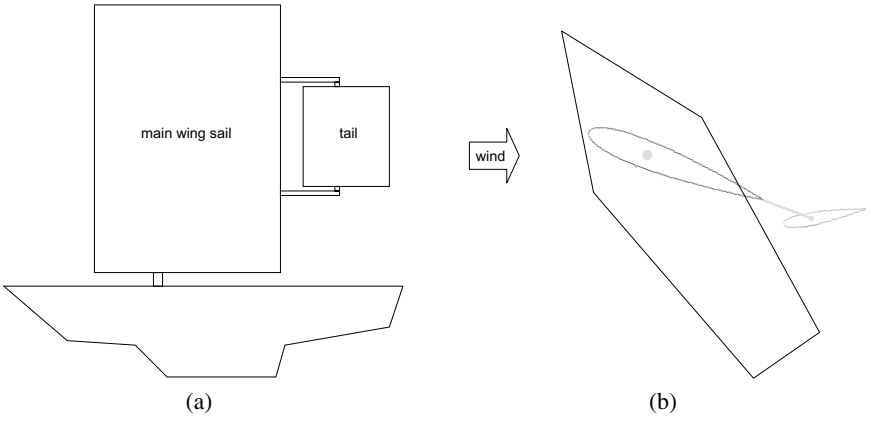


Fig. 3 Self-trimming wing sail: (a) side view of an arrangement with main wing sail and tail (b) orientation of wing sail and tail on a close hauled course

true wind speed under wind speeds from $12-25$ kn (approximately $6-13$ m/s) using a self-trimming wing sail on a 9.1 m catamaran.

3 Scientific Community and Events

3.1 Early Examples

Prior to 2005 when the idea of *Microtransat Challenge*[7] initiated a new era of collaborative research in robotic sailing, a large number of autonomous underwater vehicles (AUV) have been developed [11, 4]. However, research on autonomous surface vehicles (ASV), also known as autonomous surface crafts (ASC), was still in its early stage and mainly focused on motorised vessels [19, 28]. Just a few researchers worked on fully autonomous sailing robots. According to their publications these teams seemed not to be well linked to each other. A few of the most noticeable early examples are described briefly here.

3.1.1 Station Keeping Autonomous Mobile Platform (SKAMP)

The first attempt to autonomous sailing recorded in the literature is a project named *SKAMP (Station Keeping Autonomous Mobile Platform)*. The *SKAMP* was a wind propelled mobile surveillance platform and utilized a curving ring-shaped rigid wing sail (Figure 4(a)). It was developed in 1968 by E. W. Schieben with the Radio Corporation of America and was optimised for autonomous station keeping rather than for dynamic performance [42, 45]. Actual sailing data have never been published, so it remains unclear whether SKAMP ever sailed autonomously [21].

[7] `http://www.microtransat.org`

(a) (b) (c)

Fig. 4 Early robotic sailing boats: (a) SKAMP – Station Keeping Autonomous Mobile Platform [45] (b) RelationShip [27] (c) Atlantis

3.1.2 RelationShip

The second published autonomous sailing attempt was the *RelationShip* project of University of Applied Science in Furtwangen, Germany (Figure 4(b)). The project started in 1995 with the ambitious plan to sail around the world with an unmanned trimaran. According to Elkaim [21] the initial intention was to sail autonomously. However, after some difficulties the project changed to a remote control via satellite. After some years the project was cancelled due to regulatory difficulties. They did not get the permission to circumnavigate the globe with their unmanned RelationShip. The idea to declare the boat as floatsam did not convince the maritime authorities [47].

3.1.3 Fuzzy Logic Controlled Sailing Boat by Abril et al. [1]

The first documented results of fully autonomous sailing have been published by Abril et al. [1]. They presented a fuzzy logic controller for the rudder and sail of a sailing boat. Test runs have been carried out on a yacht model with an overall length of 1.03 m, a displacement of 4.5 kg and a sail area of 36.6 dm^2.

3.1.4 Atlantis

The *Atlantis* project of Stanford University began in 1997 with the concept of an unmanned, autonomous, GPS guided, wing-sail propelled sailing boat. The boat is based on a Prindle-19 Catamaran, with a self-trimming wing-sail (Figure 4(c)[8]). The maiden voyage took place in Redwood City Harbour in January 2001. [21, 22].

[8] ©Gabriel Elkaim.

3.2 Competitions in Robotic Sailing

In many fields of robotics competitions with memorable goals cause the attention of media and the interested public, and therefore give a strong incentive to research and development in the particular area. The most popular examples in robotics are DARPA Grand Challenge for completely autonomous cars [50] or RoboCup soccer robots which aim to beat the human world champions by 2050 [38]. The same happened to autonomous sailing during the first decade of the 21st century, when different events have been founded almost at the same time.

3.2.1 Microtransat

Research in autonomous sailing has been recently stimulated by the *Microtransat* idea of Yves Briere (ISAE, France) and Mark Neal (Aberystwyth University, Wales, UK) [15, 14]. The organizers describe the *Microtransat* in on their web site [29] as follows: *"The Microtransat Challenge is a transatlantic race of fully autonomous sailing boats. The race aims to stimulate the development of autonomous sailing boats through friendly competition."* Participating sailing boats shall further be small (max. 4 m in length), unmanned, and use wind as the only form of propulsion. A few smaller Microtransat competitions took place prior to the real transatlantic race. These allowed contestants to exchange ideas and test their boats in less harsh environments.

The first transatlantic attempt was in 2010. The only boat which entered the competition was *Pinta* from Aberystwyth University, Wales, UK. According to Colin Sauzé the boat was 49 *h* and 87 *km* under autonomous control before the computer system failed[9].

3.2.2 SailBot

SailBot is an international competition for autonomously controlled sailboats. Aimed primarily at undergraduate student teams, the goal is to give engineering students a practical application of the topics they have learned, while also providing a fun way to learn project management in a multidisciplinary environment. A successful *SailBot* balances the needs of naval architecture, mechanical engineering, systems and electrical engineering, as well as project management.[10] *SailBot* competitions were held annually since 2006, except 2007.

SailBot is open for semi-autonomous and fully autonomous sailing boats up to 2 m in length. Since 2010 they provide an additional "open class" for boats with a maximum length of 4 m. For the 2011 competition five different events are announced: (1) fleet racing (2) station keeping (3) endurance contest (4) autonomous navigation and (5) presentation and design.

[9] Mail from Colin Sauzé on Microtransat mailing list from 02.10.2010.

[10] Web site of SailBot 2011: `http://www.sname.org/SNAME/SailBot2011/Home/Default.aspx`

3.2.3 World Robotic Sailing Championship and International Robotic Sailing Conference

The World Robotic Sailing Championship (WRSC)[11] is an international competition for autonomous sailing boats. The first WRSC was held in Austria in 2008. Since then it took place every year: Portugal (2009), Canada (2010) and Germany (2011). The competition is open to boats of up to 4 *m* in length. Detailed rules of WRSC change every year to respond to recent scientific developments and stimulate certain areas of research. By keeping a rather low entry threshold, WRSC not only appeals to experts but also provides a platform for new teams in this recent field of research.

The competitions coincide with the International Robotic Sailing Conference (IRSC). This conference is the basis of scientific exchange in the robotic sailing community. The combination of competitions on the water and a scientific conference provides an opportunity to practically demonstrate theoretical developments.

3.3 Competing Teams and Their Sailing Robots

This section provides an overview of the teams which participated in recent robotic sailing competitions and are covered by scientific literature. Many of them have been encouraged by these events to start research in the field of autonomous sailing. Figure 5 shows the increasing number of boats competing in autonomous sailing competitions since their invention in 2006. The teams are presented here in alphabetical order.

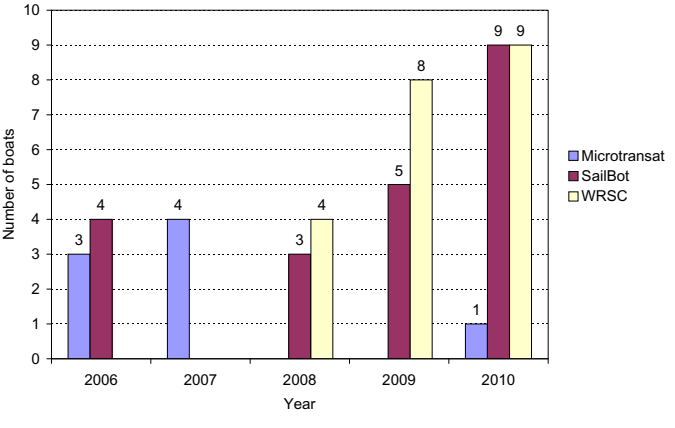


Fig. 5 Number of boats competing in Microtransat, SailBot and WRSC. In 2010 SailBot and WRSC were organised as a single event.

[11] http://www.roboticsailing.org

(a)

(b)

(c)

(d)

(e)

(f)

(g)

Fig. 6 Autonomous sailing vessels with a length of less than 2 m: (a) Däumling, University of Lübeck (b) MOOP, University of Aberystwyth (c) Pi-mal-Daumen, University of Lübeck (d) Breizh Spirit, ENSTA Bretagne (e) Roboat I, INNOC (f) AROO, University of Aberystwyth (g) ARC, University of Aberystwyth

3.3.1 Austrian Society for Innovative Computer Sciences (INNOC)

The team of INNOC focuses on control algorithms rather than on boat design. So far, two commercially available hulls have been adapted for the purpose of autonomous sailing.

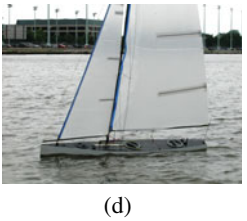
Roboat I (Figure 6(e)[12]) is based on a ready-made yacht model of type *Robbe Atlantis* intended to be remote-controlled. It is a gaff-rigged schooner[13] with a length of 1.38 m, a height of 1.73 m and a total displacement of 17.5 kg. The four sails comprise a total area of 85.5 dm^2. The boat is equiped with a 800 MHz PC running Linux, a GPS receiver, a tilt-compensated electronic compass and sensors for wind speed and wind direction. *Roboat I* won the Microtransat competition in 2006.

The basis for the *ASV Roboat* (Figure 8(e)[14]) is the commercially available boat tyle Laerling[15]. The boat was originally created for kids to learn sailing, and therefore safety and stability are its major characteristics. It has a length of 3.75 m and comprises a 60 kg keel-ballast, which will bring the boat upright even from the most

(a)

(b)

(c)



(d)

Fig. 7 Autonomous sailing vessels with a length of exactly 2 m: (a) Black Adder, Queen's University (b) First Time, USNA (c) Gill the Boat, USNA (d) Luce Canon, USNA

[12] ©INNOC.

[13] A schooner is a type of sailing vessel characterized by the use of two or more masts with the forward mast being no taller than the rear masts.

[14] ©INNOC.

[15] `http://www.laerling.nl`

Fig. 8 Autonomous sailing vessels with a length of more than 2 m: (a) IBoat, ISAE (b) FASt, University of Porto (c) Pinta, University of Aberystwyth (d) Beagle-B, University of Aberystwyth (e) ASV Roboat, INNOC (f) Avalon, ETH Zurich

severe heeling. Including batteries the overall weight of the boat is about 300 kg. The sail area of mainsail and foresail together is 4.5 m^2. It is equipped with solar panels providing up to 285 W_p of power during conditions of full sun and a direct methanol fuel cell delivering 65 W as a backup energy source. The *ASV Roboat* features a three-stage communication system, combining WLAN, UMTS/GPRS and an IRIDIUM satellite communication system, allowing continuous real-time access from shore [48]. Control software runs on a Linux-based on-board computer system using incoming data from various sensors (GPS, compass, anemometer, etc.) on an NMEA2000-bus. *ASV Roboat* won the Microtransat 2007 as well as the WRSC in 2008, 2009, and 2010.

3.3.2 École Nationale Supérieure de Techniques Avancées (ENSTA) de Bretagne

A team of ENSTA Bretagne participated with their boat *Breizh Spirit* (Figure 6(d)[16]) in WRSC 2009. Their design uses a custom built hull based on the IMOCA class design with a length of 1.3 m and two traditional sails. The control system is implemented on a PIC18F2550 microcontroller. [44]

3.3.3 Institut Supérieur de l'aéronautique et de l'espace (ISAE)

IBoat from ISAE, France (Figure 8(a)[17]) is made of fibreglass and carbon and therefore is relatively lightweight. *IBoat* has a length of 2.4 m and a height of 3 m. It features 1.5 m^2 of sail area in a combination of two sails (main sail and jib) both mounted on a balanced rig. Used sensors are an electronic compass, a wind sensor (speed and direction) and a GPS receiver. The sensors are connected to a microcontroller via CAN bus [14]. *IBoat* competed in Microtransat 2006 and 2007 as well as in WRSC 2009.

3.3.4 Queen's University

Mostly Autonomous Sailboat Team (MAST) was founded in 2004 at Queen's university. Their first vessel entering SailBot 2007 was *Black Adder* (Figure 7(a)[18]), a 2 m long carbon fibre hull with traditional sails using a PBasic Stamp for the control system. Since 2007 the team of undergraduate students made significant modifications to their first design and participated in the Microtransat Challenge 2007, WRSC 2008 and 2010 as well as SailBot 2008, 2009 and 2010. [17]

3.3.5 Swiss Federal Institute of Technology Zurich (ETH)

Avalon (Figure 8(f)[19]) was developed by a team of students from the Federal Institute of Technology, Zurich for the Microtransat challenge and participated in WRSC 2009. *Avalon* features a monohull design with a length of 3.95 m, a balanced rig and a twin rudder system. The power supply is realized with four solar panels of 90 W_p, four lithium-manganese batteries of 600 Wh each and a direct-methanol fuel cell for back-up power. The control system is implemented on an MPC21[20] industrial computer running Linux. [25]

3.3.6 United States Coast Guard Academy (USCGA)

Intuition was developed by the United States Coast Guard Academy and participated in SailBot/WRSC 2010. The USCGA's 2 m monohull design features a conventional

[16] ©Jan Sliwka.
[17] ©INNOC.
[18] ©INNOC.
[19] ©Patrick Moser.
[20] `http://www.kontron.com`

sloop rig with a sail area of 1.7 m^2. The control system is implemented on an ISIS PC104 single board computer running Windows XPe and MATLAB. [17]

3.3.7 United States Naval Academy (USNA)

USNA began their activities in 2007. They participated with their boat *First Time* (Figure 7(b)[21]) in SailBot 2008. In 2009 they entered *Luce Canon* (Figure 7(d)[22]) in SailBot and WRSC. In 2010 they participated with *Gill the Boat* (Figure 7(c)[23]) in SailBot/WRSC. The USNA team comprising undergraduate Naval Architects and Systems Engineers designs and builds new sailing vessels each year by continuously improving upon previous year's boats. Their designs use a custom built single hull with a length of 2 m and a conventional sloop rig with a sail area of about 3 m^2. [31, 30]

3.3.8 University of Aberystwyth

The team around Mark Neal and Colin Sauzé built multiple sailing robots varying in length between 50 cm and 3.5 m with the aim of performing long term autonomous missions for oceanographic monitoring.

AROO (Figure 6(f)[24]) was constructed in late 2004 as a proof of concept for a small but durable sailing robot. The hull is about 1.5 m long and is rigged with a 1 m high wing sail. [34]

ARC (Figure 6(g)[25]) is about 1.5 m in length and features two independently controlled wing sails and two rudders controlled by a single actuator. It is equipped with a gimbaled compass, GPS receiver and a combination of an AtMega128 microcotroller and a Gumstix[26] single board computer running Linux. The only power source is a bank of 20 pieces of 1.2 V AA size recharchable batteries with a capacity of 2500 mAh each. [40]

Beagle-B (Figure 8(d)[27]) is their largest boat and was constructed in late 2006 by Robosoft (a French robotics company). It is 3.5 m long and uses a 3 m solid wing sail. *Beagle-B* is intended to provide a serious oceanography platform for long term missions. Its power is provided by two 15 W solar panels and four 60 Ah batteries with 12 V. It includes a YSI 660 Sonde for gathering oceanographic data as well as an Iridium SBD transceiver and GSM modem for data transmission. *Beagle-B* participated in the Microtransat Challenge 2007 in which it sailed a total of 25 km over 19 hours. [40]

[21] ©Paul Miller.
[22] ©Paul Miller.
[23] ©Paul Miller.
[24] ©Colin Sauzé.
[25] ©INNOC.
[26] `http://www.gumstix.com`
[27] ©Colin Sauzé.

Pinta (Figure 8(c)[28]) was built for WRSC 2008 and the Microtransat transatlantic race. Unlike the other boats of the team it uses a single traditional sail controlled by an electric winch system. *Pinta* is based on a Toper Taz sailing dinghy with a length of 2.95 m. The rudder is controlled by an off the shelf auto-helm. *Pinta* was the only boat to enter the 2010 Microtransat Challenge.

The *MOOP* (Mini Ocean Observation Platform; Figure 6(b)[29]) is a small lightweight sailing robot with an overall length of 0.72 m. Multiple *MOOPs* have been built so far with single or twin wing sail configuration, with rudders and rudderless. They are controlled either solely by a PIC microcontroller or with a combination of PIC and Gumstix Single Board Computer. Several *MOOPs* took part in the SailBot and WRSC competitions in 2009 and 2010. [17]

3.3.9 University of Lübeck

University of Lübeck started their autonomous sailing activities in 2009. They participated in Sailbot/WRSC 2010 with two boats.

Däumling (Figure 6(a)[30]) is based on a University Club (Graupner, Germany) boat model with a length of 0.53 m and two sails with a total sail area of 0.145 m^2. The boat is inteded to be used as a testbed for various methods of artificial intelligence. [16]

Pi-mal-Daumen (Figure 6(c)[31]) is based on a standard IOM (International One Meter class) hull with a length of 1 m and two sails with a total sail area of 0.4 m^2. On board control is realised with a custom built circuit board featuring an ATMega2560 microprocessor. [6]

3.3.10 University of Porto

Team *FASt* (Figure 8(b)[32]) developed a 2.5 m long custom built hull and has a total sail area of 3.7 m^2. The design was inspired by modern racing ocean yachts. The control system is implemented on a small FPGA-based single board computer, including a 32-bit RISC microprocessor running at 50 MHz. Communication with the boat is possible using Wi-Fi, GSM, Iridium SBD and a conventional RC receiver used in radio-controlled models. *FASt* entered in WRSC 2008 and 2009. [5]

4 Potential Applications

Recent events, like the devastating tsunami in Asia in 2004, the Deepwater Horizon oil spill in Gulf of Mexico in 2010, accidents with refugee boats off the coast of Lampedusa, Italy, and pirate activities in the Gulf of Aden have emphasized

[28] ©Colin Sauzé.
[29] ©Colin Sauzé.
[30] ©Alexander Schlaefer.
[31] ©Alexander Schlaefer.
[32] ©José Carlos Alves.

impressively the importance of a fully integrated ocean observation system [39]. AUVs and motorised ASVs are widely-used for ocean observations since many years [10, 39]. According to Cruz and Alves [20] the main strengths for unmanned autonomous sailing boat for this task are:

- Long mission ranges
- Negligible operational costs
- Potential for towing sensors
- Real time data transmission
- Real time localisation
- Very low noise generation

Not all of the possible applications mentioned in the list below are likely to be realised in the next few years. The current focus is clearly set on ocean monitoring. But some more tasks are possible to be fulfilled by manned or unmanned sailing robots:

- Intelligent sensor buoys
- CO_2-neutral transportation of goods
- Reconnaissance and Surveillance
- Supply Vessel
- Unmanned ferrying
- Minefield mapping

Acknowledgements. This paper was written as part of *AAS Endurance*, a joint research project of INNOC, Austria and Oregon State University, USA. The aim of the project is to develop an autonomous sailing boat for passive acoustic monitoring of marine mammals and mitigation of human impacts on them. The project is realised within the funding programme *Sparkling Science*, supported by the Austrian Federal Ministry of Science and Research.

References

1. Abril, J., Salom, J., Calvo, O.: Fuzzy control of a sailboat. International Journal of Approximate Reasoning 16(3-4), 359–375 (1997), http://mapp1.de.unifi.it/persone/Allotta/ICAD/Abril1997.pdf
2. Adriaans, P.W.: From knowledge-based to skill-based systems: Sailing as a machine learning challenge. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 1–8. Springer, Heidelberg (2003), http://www.springerlink.com/content/9fl801mf9pkwdgdy/
3. Allensworth, T.: A short history of Sperry Marine (1999), http://www.sperrymarine.northropgrumman.com/Company-Information/Corporate-History/Sperry-History/
4. von Alt, C.: Autonomous underwater vehicles. In: Autonomous Underwater Lagrangian Platforms and Sensors Workshop (2003), http://www.geo-prose.com/ALPS/white_papers/alt.pdf

5. Alves, J., Ramos, T., Cruz, N.: A reconfigurable computing system for an autonomous sailboat. In: IRSC 2008 International Robotic Sailing Conference, pp. 13–20 (2008), http://www.roboticsailing.org/fileadmin/user_upload/_temp_/Roboticsailing/Proceedings-web.pdf#page=13
6. Ammann, N., Hartmann, F., Jauer, P., Bruder, R., Schlaefer, A.: Design of a robotic sailboat for wrsc/sailbot. In: International Robotic Sailing Conference 2010, pp. 40–42 (2010)
7. BalancedRig: Balancedrig website (2009), http://balancedrig.com/description.html (accessed April 16, 2009)
8. Benatar, N., Qadir, O., Owen, J., Baxter, P., Neal, M.: P-Controller as an Expert System for Manoeuvring Rudderless Sail Boats. In: Proceedings of UKCI (2009), http://www-users.york.ac.uk/~oq500/pdfs/UKCI09-paper.pdf
9. Bennett, S.: A history of control engineering, 1800-1930. Inspec/Iee (1986)
10. Bertram, V.: Unmanned surface vehicles–a survey. Skibsteknisk Selskab, Copenhagen, Denmark (2008), http://www.skibsteknikselskab.dk/public/dokumenter/Skibsteknisk/Download20materiale/2008/1020marts2008/USVsurvey_DTU.pdf
11. Blidberg, D.: The development of autonomous underwater vehicles (auvs); a brief summary. In: IEEE ICRA, vol. 4, Citeseer (2001), http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.122. 1739&rep=rep1&type=pdf
12. Bohnenberger, J.G.F.: Beschreibung einer Maschine zur Erläuterung der Gesetze der Umdrehung der Erde um ihre Axe, und der Veränderung der Lage der letzteren. Tübinger Blätter für Naturwissenschaften und Arzneikunde 3, 72–83 (1817), http://www.ion.org/museum/files/File_1.pdf
13. Bowditch, N.: The American Practical Navigator. Paradise Cay Publications (2010), http://amazon.com/o/ASIN/0939837544/
14. Briere, Y.: Iboat: An autonomous robot for long-term offshore operation. In: The 14th IEEE Mediterranean Electrotechnical Conference, MELECON 2008, pp. 323–329. IEEE, Los Alamitos (2008), ftp://www.inc.eng.kmutt.ac.th/pornpoj/SailBoat/getPDF.pdf
15. Briere, Y., Bastianelli, F., Gagneul, M., Cormerais, P.: Challenge Microtransat. In: CETSIS 2005, Nancy, France (2005), http://oatao.univ-toulouse.fr/126/1/Briere_126.pdf
16. Bruder, R., Stender, B., Schlaefer, A.: Model sailboats as a testbed for artificial intelligence methods. In: Proceedings of the 2nd International Robotic Sailing Conference, pp. 37–42 (2009)
17. Burnie, M. (ed.): Participant Package – World Robotic Sailing Championship 2010 and International Robotic Sailing Conference 2010. Queen's University, Kingston (2010)
18. Burns, R.: The use of artificial neural networks for the intelligent optimal control of surface ships. IEEE Journal of Oceanic Engineering 20(1), 65–72 (1995), http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=380245
19. Caccia, M.: Autonomous Surface Craft: prototypes and basic research issues. In: 14th Mediterranean Conference on Control and Automation, MED 2006, pp. 1–6. IEEE, Los Alamitos (2006), http://www.engr.mun.ca/~bachmayer/ENG9095-webpage/ASC_USV/ASC_survey_paper2006.pdf
20. Cruz, N., Alves, J.: Ocean sampling and surveillance using autonomous sailboats. In: IRSC 2008 International Robotic Sailing Conference, p. 30 (2008), http://innoc.at/fileadmin/user_upload/_temp_/Roboticsailing/Proceedings-web.pdf#page=30

21. Elkaim, G.: System identification for precision control of a wingsailed GPS-guided catamaran. Ph.D. thesis, Standford University (2002)
22. Elkaim, G.: The Atlantis project: A GPS-guided wing-sailed autonomous catamaran. Navigation 53(4) (2006), http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.153. 9098&rep=rep1&type=pdf
23. Elkaim, G., Boyce, L.C.O.: Experimental Aerodynamic Performance of a SelfTrimming Wing-Sail for Autonomous Surface Vehicles. In: Proc. Of the IFAC Conference on Control Applications in Marine Systems, IFAC CAMS. Citeseer (2007), http://users.soe.ucsc.edu/~elkaim/Documents/PerfCAMS07.pdf
24. Enab, Y.: Intelligent controller design for the ship steering problem. In: IEE Proceedings Control Theory and Applications, vol. 143(1), pp. 17–24. IET (1996), http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=487979
25. Giger, L., Wismer, S., Boehl, S., Büsser, G., Erckens, H., Weber, J., Moser, P., Schwizer, P., Pradalier, C., Siegwart, R.: Design and construction of the autonomous sailing vessel avalon. In: Proc. 2nd Int. Robotic Sailing Conf., pp. 17–22 (2009)
26. Layne, J., Passino, K.: Fuzzy model reference learning control for cargo ship steering. IEEE Control Systems Magazine 13(6), 23–34 (1993), http://www2.ece.ohio-state.edu/~passino/PapersToPost/FMRLC-ship.pdf
27. Loibner, D.: KLAR zur WENDE? klick. Yacht Revue 3 (1998), http://www.esys.org/stories/trimaran.html
28. Manley, J.: Unmanned surface vehicles, 15 years of development. In: OCEANS 2008, pp. 1–4. IEEE, Los Alamitos (2008), http://www.oceanicengineering.org/history/080515-175.pdf
29. Microtransat: Official microtransat web site (2011), http://www.microtransat.org (accessed on 27 April 2011)
30. Miller, P., Beal, B., Capron, C., Gawboy, R., Mallory, P., Ness, C., Petrosik, R.: Increasing performance and added capabilities of usna sail-powered autonomous surface vessels (asv). In: International Robotic Sailing Conference (2010), http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA534798&Location=U2&doc=GetTRDoc.pdf
31. Miller, P., Brooks, O., Hamlet, M.: Development of the usna sailbots (asv). In: International Robotic Sailing Conference (2009), http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA534672&Location=U2&doc=GetTRDoc.pdf
32. Minorsky, N.: Directional stability of automatic steered bodies. Journal of American Society of Naval Engineers 34(2), 280–309 (1922)
33. Multirig: Multirig website (2009), http://www.multirig.com/the_balestron_rig.htm (accessed April 16, 2009)
34. Neal, M.: A hardware proof of concept of a sailing robot for ocean observation. IEEE Journal of Oceanic Engineering 31(2), 462–469 (2006), http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.103. 1486&rep=rep1&type=pdf
35. Neal, M., Sauze, C., Thomas, B., Alves, J.: Technologies for Autonomous Sailing: Wings and Wind Sensors. In: Proceedings of the 2nd IRSC, Matosinhos, Portugal, July 6-12, pp. 23–30 (2009), http://cadair.aber.ac.uk/dspace/bitstream/2160/3103/2/IRSC.pdf
36. Polkinghorne, M., Roberts, G., Burns, R., Winwood, D.: The implementation of fixed rulebase fuzzy logic to the control of small surface ships. Control Engineering Practice 3(3), 321–328 (1995), http://202.114.89.60/resource/pdf/2231.pdf
37. Roberts, G.: Trends in marine control systems. Annual reviews in control 32(2), 263–269 (2008), http://202.114.89.60/resource/pdf/2236.pdf

38. RoboCup: Official robocup website: Objective (2011), http://www.robocup.org/about-robocup/objective/ (accessed May 1, 2011)
39. Rynne, P., von Ellenrieder, K.: Unmanned autonomous sailing: Current status and future role in sustained ocean observations. Marine Technology Society Journal 43(1), 21–30 (2009)
40. Sauzé, C., Neal, M.: Design considerations for sailing robots performing long term autonomous oceanography. In: Proceedings of The International Robotic Sailing Conference, May 23 -24, pp. 21–29 (2008), http://www.innoc.at/fileadmin/user_upload/_temp_/Roboticsailing/Proceedings-web.pdf
41. Scanmar: Classification of Vane Gears by Course Correcting System (2011), http://www.selfsteer.com/windvanes101/classification.php (accessed on April 27, 2011)
42. Schieben, E.W.: Skamp – an amazing unmanned sailboat! Ocean Industry, pp. 38–43 (1969)
43. Shukla, P., Ghosh, K.: Revival of the Modern Wing Sails for the Propulsion of Commercial Ships. International Journal of Civil and Environmental Engineering, 75–80 (2009), http://www.akademik.unsri.ac.id/download/journal/files/waset/v1-2-14-19.pdf
44. Sliwka, J., Reilhac, P., Leloup, R., Crepier, P., Malet, H., Sittaramane, P., Bars, F., Roncin, K., Aizier, B., Jaulin, L.: Autonomous robotic boat of ensieta. In: 2nd International Robotic Sailing Conference, Matosinhos, Portugal (2009), http://media.ensta-bretagne.fr/robotics/images/4/4f/Ensieta_team.pdf
45. Smith, B.: Skamp – roboat boat with rigid sails patrols ocean beat. Popular Science 196(5), 70–72 (1970), http://books.google.at/books?id=8QAAAAAAMBAJ&pg=PA70&dq=SKAMP+Schlieben&source=gbs_toc_r&cad=2#v=onepage&q=SKAMP20Schlieben&f=false
46. Sperry, E.: Automatic steering. Society of Naval Architects and Marine Engineers (1922)
47. Spiegel: Fliegender Schwarzwälder. Der Spiegel 22, 176–177 (1998), http://wissen.spiegel.de/wissen/image/show.html?did=7894363&aref=image017/SP1998/022/SP199802201760177.pdf&thumb=false
48. Stelzer, R., Jafarmadar, K.: Communication architecture for autonomous sailboats. In: Proceedings of International Robotic Sailing Conference, Matosinhos, Portugal, pp. 31–36 (2009), ftp://www.inc.eng.kmutt.ac.th/pornpoj/SailBoat/1266686522.pdf
49. Stelzer, R., Pröll, T., John, R.: Fuzzy logic control system for autonomous sailboats. In: FUZZ-IEEE 2007, London, UK, pp. 97–102 (2007), http://dx.doi.org/10.1109/FUZZY.2007.4295347
50. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., et al.: Stanley: The robot that won the DARPA Grand Challenge. In: The 2005 DARPA Grand Challenge, pp. 1–43 (2007), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.1920&rep=rep1&type=pdf
51. Van Aartrijk, M., Tagliola, C., Adriaans, P.: AI on the Ocean: the RoboSail Project. In: ECAI, pp. 653–657. Citeseer (2002), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.8172&rep=rep1&type=pdf
52. Van Amerongen, J.: Adaptive steering of ships–A model reference approach. Automatica 20(1), 3–14 (1984), http://www.ce.utwente.nl/rtweb/publications/1984/pdf-files/Automatica84.pdf

53. Velagic, J., Vukic, Z., Omerdic, E.: Adaptive fuzzy ship autopilot for track-keeping. Control engineering practice 11(4), 433–443 (2003), http://202.114.89.60/resource/pdf/2235.pdf
54. Worsley, P.: Self Trimming/Self Tending Wingsails (2011), http://www.ayrs.org/wingsails.pdf (accessed on May 8, 2011)
55. Yeh, E., Bin, J.: Fuzzy control for self-steering of a sailboat. In: Proceedings of Singapore International Conference on Intelligent Control and Instrumentation, SICICI 1992, vol. 2, pp. 1339–1344. IEEE, Los Alamitos (1992), http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=637738
56. Zirilli, A., Tiano, A., Roberts, G., Sutton, R.: Fuzzy course-keeping autopilot for ships. In: 15th Triennial World Congress, Barcelona, Spain (2002), http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac2002data/content/00824/824.pdf

Part II
Robotic Sailboats

Sailing without Wind Sensor and Other Hardware and Software Innovations

Jan Sliwka, Jeremy Nicola, Remi Coquelin, Francois Becket de Megille, Benoit Clement, and Luc Jaulin

Abstract. In our paper, we propose a solution for a robotic sailboat which is adapted to long travel across the ocean. We tried to be as innovative as possible and stray out of the classical designs. In fact, robots often have different shapes from vehicles for mankind since first of all there are no human on board which means less constraints with regards to security, ergonomy or comfort. In this paper two innovations are presented. The first one addresses the main issue when navigating for a long time with a fully actuated sailing robot which is the energy used for guidance, navigation and control. One can use big solar panels to overcome that problem. Instead, we propose to use a wind vane self steering device. Our idea is to put this device on the bow which simplifies its design. This device regulates the trajectory of the boat relatively to wind direction. As a consequence, the wind sensor is not necessary. The second innovation is a strategy of navigation between waypoints. This paper will also present the prototype we used for testing those algorithms.

1 Introduction

Since many years unmanned surface vehicles (USVs) has been used for applications such as oceanography, harbour protection or military (as weapons or moving targets) [2]. USVs generally use propellers to move an get their energy from fossil fuels or solar power. It is only recently that USVs use sail as a mean of propulsion [1] and also became autonomous. The use of such vehicles is especially needed in oceanography [7][8][9] where there is a need for several months of data acquisition in a designed area. In fact the currently used buoys use to drift away very quickly from the data acquisition area and are often impossible to moor because of ocean's depth. Research in autonomous sailing boats was also stimulated by the

Jan Sliwka · Jeremy Nicola · Remi Coquelin · Francois Becket de Megille · Benoit Clement · Luc Jaulin
ENSTA Bretagne, 2 rue Francois Verny, 29200 Brest, France
e-mail: {jan.sliwka,jeremy.nicola}@ensta-bretagne.fr

MicroTransat challenge [3] which purpose is to cross the Atlantic ocean in full autonomy. Many autonomous sailing robots builders tend to make copies of existing sailboats for mankind installing the same set of sensors and replacing human actuation by electronic actuators. ENSTA-Bretagne robotic team has chosen to work on sailing robots capable of navigation for long periods. In order to reduce the energy consumption and to improve the robustness of the robot, we focus on reducing the number of sensors and actuators (especially the wind sensor) and develop new algorithms and/or mechanical designs to deal with that handicap. In this paper, we propose a solution for a robotic sailboat which addresses the main issue when navigating for a long time with a fully actuated sailing robot which is the energy used for guidance, navigation and control. One can use big solar panels to overcome that problem. Instead, we propose the use of a wind vane self steering device. This kind of device actually exists on real boats [4], but our approach is to install this device on the bow instead of putting it on the stern since it greatly simplifies its design. A mechanical design proposition will be provided for this device. Our first aim is to validate trajectory stabilization by the device. The final purpose is to sail using only the self steering device (without any rudder on the stern). Sailing using the self-steering device enables navigation relatively to the wind. As a consequence, there is no need for the wind sensor. Simulations are provided for navigation maneuvers such as tacking and jibbing for both cases when the regulator is on the bow and the classical case when it is on the stern.

In this paper we also present algorithms for navigation between waypoints. There are many different methods which can be used for autonomous navigation [11] [6]. Our algorithm allows the boat to navigate on straight lines between waypoints taking the drift of the boat into consideration. Simulations results are provided to prove the usefulness of the approach.

Finally, we will present our test prototype which is used for testing the concepts above. The design has been driven by the simplicity in construction.

2 Sailing without Wind Sensor

2.1 Wind Rudder Regulator

According to [4] the idea that a sailing boat might be able to steer itself using wind did not appear until the twentieth century. It might be basically because the crew was cheap and was doing the job perfectly. The first model of self steering wind vane was installed on a motorboat in 1936. It was connected to the rudder by lines. Nowadays, the wind vane self steering devices are also used on sailboats. We remarked that those devices were put on the stern. In fact, a boat can often be controlled single-handedly by a human so naturally the wind vane, the rudder, sail lines are on the stern. Besides, the bow is often hardly accessible. Finally, the bow is subject to waves splash and more rough conditions. On the other hand, robot actuators can be dispatched everywhere on its body so that is when we came with the idea to put the self steering system on the bow. Figure 1 illustrate the difference between

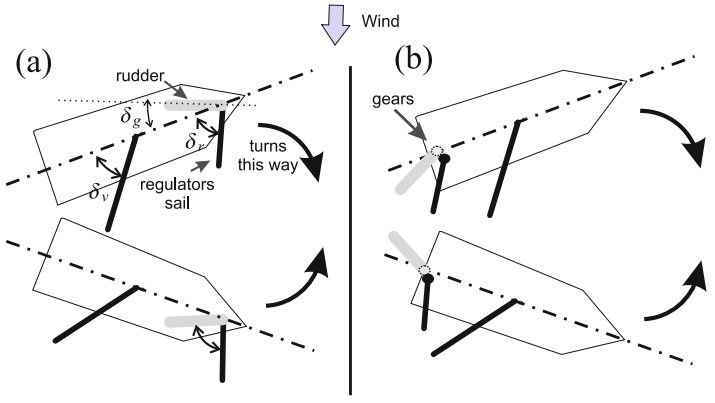


Fig. 1 Sub-Figure (b) shows a classical design of a wind vane self steering device when the rudder is on the stern. In this case there is a need to invert the rotation between the wind vane and the rudder using gears for example. Sub-Figure (a) shows the new simplified design where the rudder is on the front and the inversion of rotation is no longer necessary.

the bow and stern designs. The main difference is that there is no need for gears to invert the rotation between the wind vane and the rudder. We will remark later in the simulation part that those two designs are equivalent in terms of trajectory stabilization.

2.2 Simulation

2.2.1 Equations

Consider a dynamic system defined by the following evolution function

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{1}$$

$\mathbf{x}$ being the state of the system.

For the simulation, we use Euler approximation to estimate the state of the robot in real time as follows

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)\mathbf{dt} + \mathbf{x}_k \tag{2}$$

$\mathbf{x}_k$ being the state of the system and u_k being the command at time step k. For the simulation of the robot, we used the state equations from [5]. Figure 2 shows the block diagram of the system {boat,wind vane regulator}.

Denote by δ_r and δ_g, respectively the angle between the wind vane and the axis of the boat, and the angle between the rudder and the axis of the boat as shown in Figure 1. δ_{v0} is the sail command which will be left constant for the simulation. In fact, our prototype have an non actuated sail to add robustness. Denote by $\overrightarrow{V}$ the wind vector. Denote by ϕ the angle between the rudder and the wind vane when the

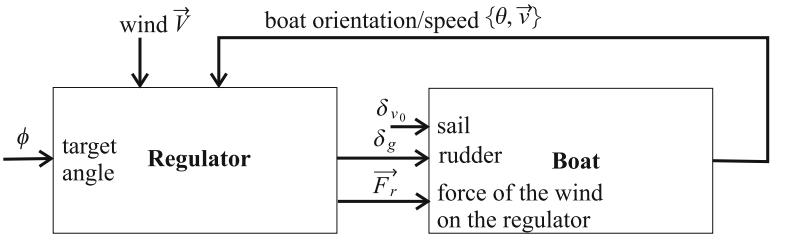


Fig. 2 This figure represents the block diagram corresponding to the system {boat,wind vane regulator}. The regulator will control the angle δ_g of the rudder of the boat and exerts a force $\mathbf{F}_r$ on the boat due to the drag of the wind vane. The regulator will be moved by the wind ($\mathbf{V}$). The angle between the rudder and the wind vane when the rudder is in zero position is determined by the variable ϕ. The sail command is left constant.

rudder is in zero position. ϕ determines the target angle to be reached by the boat with respect to wind direction.

We made some assumptions in order to simplify the model. First we assumed the rudder is perfectly compensated i.e. the force due to the water current is applied directly on the axis of the rudder. As such, the wind vane (not hindered by the torque of the rudder) will align itself with the apparent wind unless the rudder is blocked by reaching its limit angle. We also assumed that the apparent wind is equal to the real wind. This approximation is valid for slow boats such as ours. We also neglected the dynamics of the sail and wind vane since their inertia is small with regards to the magnitude of forces generated by the wind.

When the rudder is not blocked i.e. $\delta_g \in [-\delta_{g_{\max}}, \delta_{g_{\max}}]$ we have,

$$\phi = \delta_r - c \cdot \delta_g \tag{3}$$

where $c = 1$ corresponds to the case where the self steering mechanism is on the bow and $c = -1$ when it is on the stern.
The wind and the regulator are aligned thus

$$\begin{aligned} &\pi + \delta_r + \theta = \arg(\overrightarrow{V}) \\ &F_r = 0 \end{aligned} \tag{4}$$

As such

$$\delta_g = c(\arg(\overrightarrow{V}) - \pi - \theta - \phi) \tag{5}$$

When the rudder reaches its limits, it becomes blocked. In this case, we have $\delta_g = -\delta_{g_{\max}}$ or $\delta_g = \delta_{g_{\max}}$ depending on the orientation of the boat relatively to the wind and the sign of ϕ. In this case one can compute the force F_r of the wind applied to the wind vane and consequently on the boat.
We have

$$\delta_r = \phi + c \cdot \delta_g \tag{6}$$

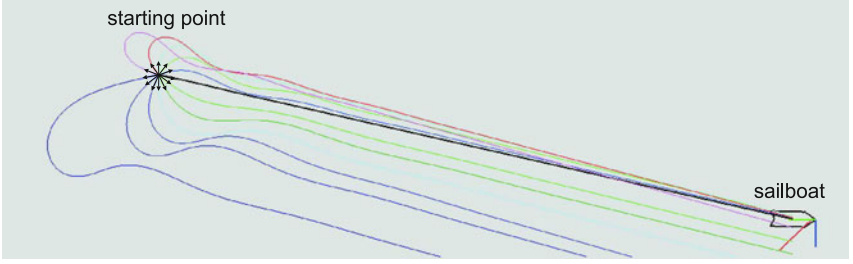


Fig. 3 We made SCILAB simulation. We launched the boat in several different directions. We observed that the boat is able to reach the desired course whatever the starting angle is which validates our approach. (simulation wise)

Consider δ_{r0} the angle of the wind vane in the global workspace.

$$\delta_{r0} = \theta + \pi + \phi + c \cdot \delta_g \tag{7}$$

As such

$$F_r = F_{r\max} \det(\overrightarrow{u_V}, \overrightarrow{u_{\delta_{r0}}}) \tag{8}$$

where $F_{r\max}$ is the force applied on the wind vane when it is perpendicular to the wind. $\overrightarrow{u_V} = \frac{\overrightarrow{V}}{\|\overrightarrow{V}\|}$ and $\overrightarrow{u_{\delta_{r0}}} = (\cos\delta_{r0}, \sin\delta_{r0})$.

2.2.2 Result of Stabilization

Figure 3 represents the trajectories computed while starting the boat at different angles. We observed that the boat is able to reach the desired course whatever the starting condition is.

2.2.3 Navigation Using Only the Self Steering Device

It is possible to navigate using only the self steering device by progressively changing the target angle ϕ. Figure 4 shows the tacking and jibbing maneuvers for both case of the self steering device. In fact, the trajectories are the same. However, in reality, the big sail will hide the wind vane of the self steering device in some of the cases. As a consequence, unless rising the wind vane above the big sail (or the opposite) the boat might have difficulties to jibe with a bow self steering device and to tack with a stern self steering device.

Note that it is necessary to have some speed to perform tacking or else the boat will be unable to cross the wind line. We are currently working on strategies to force the tacking meneuver when the speed of the boat is null. The idea is to block the rudder and use the wind vane to push the boat backwards. If the rudder is blocked in a suitable position the tacking maneuver will be accomplished.

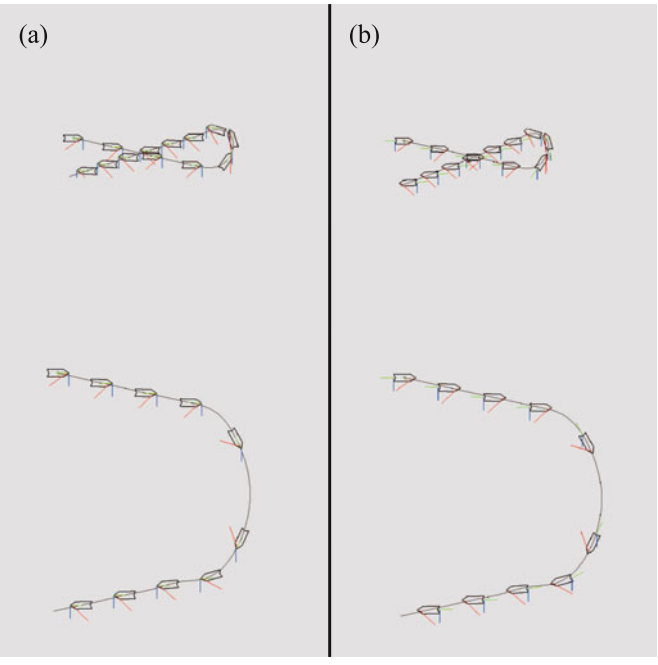


Fig. 4 Executing jibbing and tacking maneuvers using the self steering device (SCILAB simulation). Sub-Figure (a) corresponds to the case where the self steering device is on the bow. Sub-Figure (b) corresponds to the case when it is on the stern.

2.3 Mechanical Design

In this part we explain in more details the mechanical design of the bow wind vane self steering system. Installing the system on the bow simplifies the design of the system since there is no need for gears to invert the rotation of the wind vane and the rudder. As for the robustness, installing a mechanism bow put it under the action of the wave which might destroy it. However, in case of a bow self steering device the most fragile part, which is the wind vane, will align itself with the wind and the waves reducing their impact on the structure. Figure 5 represents the overall design of the mechanism.

The mechanics is really simple. However, because the robot is small, the linkage has to have less friction than the one used on big boats since the applied forces are smaller. To overcome this problem, it is possible to use bearings but because of salt corrosion, one have to use special bearings such as plastic bearings with glass balls. We propose the following design of the bow self steering system using the fact that a round buoy rotates smoothly on sea water as seen in Figure 6.

Figure 7 represents our first attempt to construct the self steering device. The buoy is constructed using extruded polystyrene covered with fiberglass and resin for

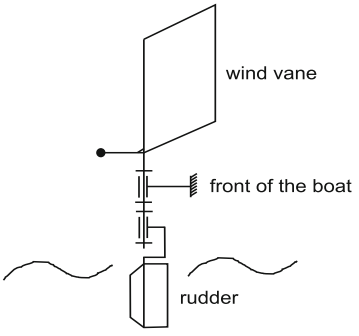


Fig. 5 The kinematic diagram of the bow wind vane self steering device

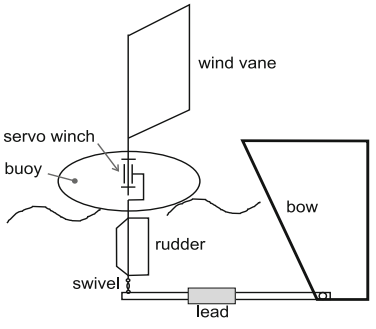


Fig. 6 Our front wind rudder regulator design

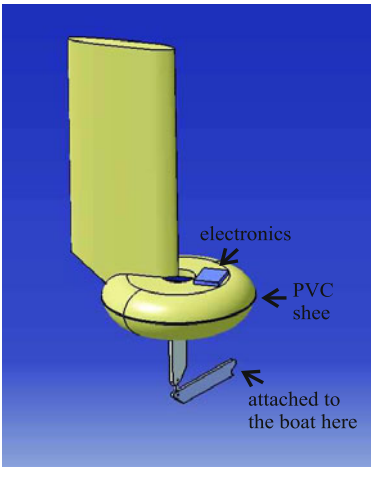


Fig. 7 The actual design of the first prototype using CATIA (computer assisted prototyping software)

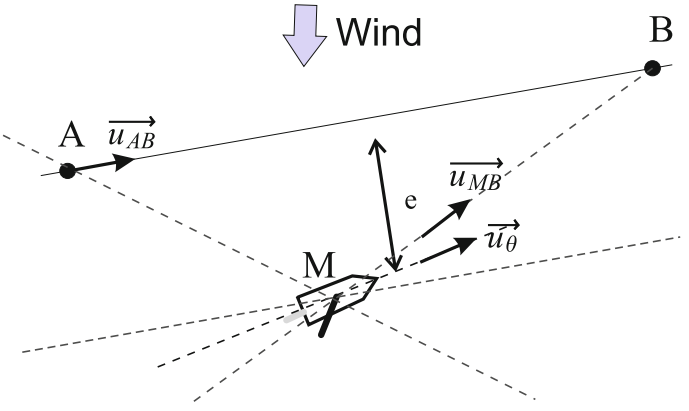


Fig. 8 This figure illustrates a sailboat navigating from a waypoint A to a waypoint B

more robustness. A PVC sheet is sandwiched between two parts of the polystyrene. The PVC sheet supports a servowinch and a box for the electronics.

3 Navigating in Straight Line

We consider a robot boat navigating from a waypoint A to waypoint B as seen in Figure 8. If the waypoint B is not upwind, the strategy is to try to regulate the boat such as it always targets waypoint B. Denote by $M = (x, y)$ the positi on of the robot and $B = (x_b, y_b)$ the position of the waypoint and θ the orientation of the robot (given by compass measurements for example). Consider the following vectors

$$\begin{aligned} \overrightarrow{u_\theta} &= (\cos\theta, \sin\theta) \\ \overrightarrow{u_{MB}} &= \frac{\overrightarrow{MB}}{\left\|\overrightarrow{MB}\right\|}. \end{aligned} \tag{9}$$

Denote by $u \in [-1, 1]$ the command associated to the rudder control. $u = 0$ means a straight rudder, 1 rotated to the max position counterclockwise and -1 rotated to the max position clockwise. The command u will be

$$u = K * \det(\overrightarrow{u_{MB}}, \overrightarrow{u_\theta}). \tag{10}$$

This strategy works but due to the drift, the robot will be facing the wind when approaching the waypoint B and eventually slow down (see trajectory (a) on Figure 9). We propose a hybrid approach which consists on following two different rules (and not one). The first rule is to follow the direction $\overrightarrow{AB}$. This rule is not sufficient to reach waypoint B because of the drift (see trajectory (c) on Figure 9). The second rule is to minimize the distance e between the boat and the line (AB). As such, the

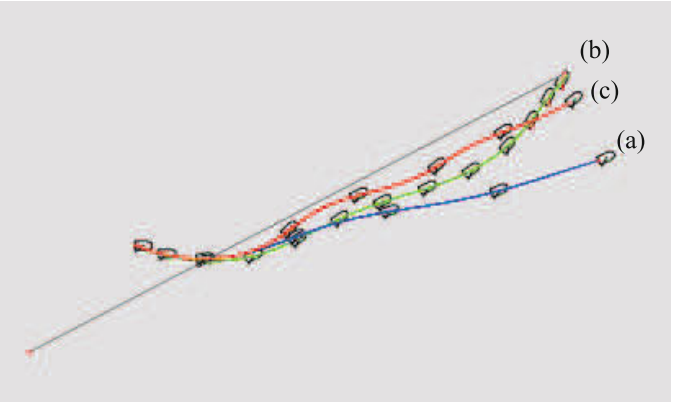


Fig. 9 Simulation results of three different algorithms

boat will have a tendency to sail parallely to the line (AB). Also, the boat will not loose speed arriving at the waypoint B.(see trajectory (c) on Figure 9)

Consider the vector

$$\overrightarrow{u_{AB}} = \frac{\overrightarrow{AB}}{\left\|\overrightarrow{AB}\right\|}. \tag{11}$$

The distance e between the robot and the line (AB) is expressed as

$$e = \det(\overrightarrow{u_{AB}}, \overrightarrow{AM}). \tag{12}$$

The basic hybrid command u can thus be defined by

$$u = K_1 * \det(\overrightarrow{u_{AB}}, \overrightarrow{u_\theta}) + K_2 * e. \tag{13}$$

In practice there are some conditions so that this formula works better. For example, we started by regulating the angle of the boat first before caring about the distance e ($K_2 = 0$ while $\det(\overrightarrow{u_{AB}}, \overrightarrow{u_\theta})$ important). For more stability, when the boat is nearly on the line (AB) we replace $K_2 * e$ by $K_2 * \arctan(K_3 * e)$.

When it is possible to have a good estimation of the speed vector (which is not always the case especially if the speed of the robot is slow), we propose a command giving similar results to the last hybrid command. The idea is to regulate the boat such as its speed vector denoted by $\overrightarrow{u_v}$ points the waypoint B. The command becomes

$$u = K * \det(\overrightarrow{u_{MB}}, \overrightarrow{u_v}). \tag{14}$$

Note that because of the drift $\overrightarrow{u_v} \neq \overrightarrow{u_\theta}$.

Fig. 10 The robot "L'improbable"

4 Our Prototype

4.1 Introduction

In 2011, two prototypes were developed at ENSTA Bretagne. One is a copy of "Breizh Spirit" [10] and the other is "L'improbable" (see Figure 10) which is based on the "Optimist" type boat design (`http://optiworld.org/`). "L'improbable" is designed to try new solutions in terms of long period ocean navigation.

4.2 Mechanical Design of "L'Improbable"

The core of the boat is an aluminum board which acts as a keel and supports the mast as seen in Figure 7. The rudder is attached to the plaque as well. Surrounding the board there is an extruded polystyrene made hull. The hull was profiled using hot wire cutting technique which is widely used in RC plane model construction. The design core board + extruded polystyrene enables fast and robust prototyping of any type of boats. The polystyrene can be made more resistant by covering it with a fiberglass layer.

We decided not to actuate the sail to increase even more the robustness of the boat. The sail is thus set at a fixed angle (an empirical angle which works the best). However, even if we loose optimality, the preliminary tests on the Ty-Colo lake (see Figure 8) showed that the boat can navigate upwind. The navigation between waypoints as shown in section 3 has also been tested using the prototype.

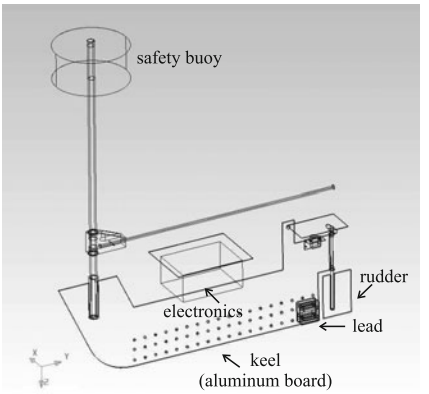


Fig. 11 The core board of "L'improbable" is an aluminum board which supports the mast, the rudder and the hull

5 Conclusion

In this paper we presented a possible solution to reduce electrical power consumption by replacing electronic boat heading correction (electronic autopilot) by a mechanical correction using a wind vane self steering device. We presented simulated results of heading stabilization and navigation maneuvers using the self steering device. Simulations look promising. The advantage of navigation using only the wind vane self steering device is that this navigation is relative to the direction of the wind. As a consequence the wind sensor is no longer necessary. In this paper we also presented few algorithms to navigate between waypoints. The real tests were performed using our recently build test platform presented in the last section.

References

1. http://www.uovehicles.com
2. Bertram, V.: Unmanned surface vehicles, a survey. Skibsteknisk Selskab, Copenhagen, Denmark (2008)
3. Briere, Y., Bastianelli, F., Gagneul, M., Cormerais, P.: Challenge microtransat. In: CETSIS 2005, Nancy, France (2007)
4. Forthmann, P.C.: Self-steering under sail. Windpilot
5. Jaulin, L.: Modelisation et commande d'un bateau a voile. In: CIFA2004 (Conference Internationale Francophone d'Automatique), CDROM, Douz, Tunisie (2004)
6. Schlaefer, A., Bruder, R., Stender, B.: Model sailboats as a testbed for artificial intelligence methods. In: IRSC 2009, pp. 37–42 (2009)
7. Sauze, C., Neal, M.: An autonomous sailing robot for ocean observation. In: TAROS, pp. 190–197 (2006)

8. Sauze, C., Neal, M.: Design considerations for sailing robots performing long term autonomous oceanography. In: IRSC 2008, pp. 21–29 (2008)
9. Sauze, C., Neal, M.: Ocean sampling and surveillance using autonomous sailboats. In: IRSC 2008, pp. 30–36 (2008)
10. Sliwka, J., Reilhac, P.H., Leloup, R., Crepier, P., Malet, H.D., Sittaramane, P., Bars, F.L., Roncin, K., Aizier, B., Jaulin, L.: Autonomous robotic boat of ensieta. In: IRSC 2009, Matosinhos, Portugal (2009)
11. Stelzer, R., Proll, T., John, R.: Fuzzy logic control system for autonomous sailboats. In: IEEE International Conference on Fuzzy Systems (2007)

MOOP: A Miniature Sailing Robot Platform

Colin Sauzé and Mark Neal

Abstract. MOOPs (Miniature Ocean Observation Platforms) are small, low cost, lightweight sailing robots intended to form a simple and flexible platform for developing robotic sailing concepts and for entering the Microtransat Challenge. They are only 72 cm long, weigh 4 kg and can be easily transported and deployed. A variety of different configurations have been tested with the intention of testing new wind sensor designs, performing experiments in autonomous power management, rudderless control and discovering wind direction without a wind sensor. Tests have been performed in lakes and at sea. During this process several limitations have been identified which cause difficulty sailing upwind, difficulty sailing downwind in single wing sail models and low hull speeds which make cause problems when fighting strong currents or tides. Future work will focus on more ocean going sailing and methods to reduce overall power consumption.

1 Introduction

This paper details the design, construction and testing of sailing robots known as MOOPs (Miniature Ocean Observation Platforms). These robots only cost around £500 (parts only, 2010 prices), are 72 cm long, weigh approximately 4 kg and follow a long keel design inspired by a Nordic folk boat. Their small size is intended to make them easy and cheap to produce in large quantities. It also enables them to be easily handled by one person, to not cause problems for other marine traffic, to be transported in a small car or as baggage on an airliner for use abroad. Their low cost enables them to be deployed in large quantities, to be lent to students, to be adapted

Colin Sauzé · Mark Neal
Department of Computer Science, Aberystwyth University
e-mail: {cos,mjn}@aber.ac.uk

for experimental purposes or to be sent on high risk missions with poor chances of recovery.

2 Boat Designs

A number of design goals were considered when the MOOPs were initially proposed. Key to these was to produce a small, cheap, robust and lightweight boat to cross the Atlantic Ocean as part of the Microtransat Challenge. It was suggested that several small, cheap and simple boats might between them stand a greater chance of one succeeding than a single larger, more expensive boat. A long keel design was selected in order to have the rudder extend from the back of the keel and reduce the possibility of seaweed and other debris from becoming entangled in it. To avoid making any holes in the hull itself a magnetic linkage was used between the rudder actuator and the rudder itself. This had the added advantage of allowing the rudder to be knocked out of alignment without risking any damage to the actuator. Previous experience [9] in building sailing robots had highlighted the advantage of wing sails. Wing sails provide a robust system that avoids the fragility of any lines connecting to the sail, the potential of winches to slip and the need to either run lines through the deck or place actuators above the deck.

After the development of an initial prototype, the potential for using these boats for researching control system design, rig design, rudderless sailing robots and autonomous power management was realised. This spawned a number of variations to the original MOOP design, including boats with twin wing sails, without rudders, without wind sensors and a variety of wind sensor designs.

With the exception of the initial prototype (MOOP0) all other MOOP hulls have been built around a mould using glass fibre and epoxy resin. A single piece perspex deck is screwed down onto the hull using 28 plastic screws. A neoprene gasket helps to ensure that the join between the deck and the hull is waterproof. A hatch in the deck, built from a plumbing screw cap is used to access the power switch, charging cable and data cables. The mast connects to a Futaba S3306MG servo located inside the hull, which is able to rotate it to adjust the sail position. It passes through several oil seals to prevent water from entering the boat through the mast. Wires run from the computer, through the mast to a female DIN plug. A tube in the centre of the sail fits over the mast and located inside it is a male DIN plug, wires then run from this plug to a wind sensor a the top of the sail. See Figure 1 for a diagram of the sail assembly.

The sail is constructed by cutting a wing shape from a block of polystyrene using a hot wire cutter, hollowing out the centre of each to make room for a carbon fibre shaft which slides over the mast and the wires which connect to the wind sensor. The hollowed out area is then filled with polystyrene balls and epoxy resin. Finally the sail is covered in glass fibre cloth and painted with epoxy resin to give it rigidity.

A combination of lead shot and epoxy resin fills the keel to provide ballast. This also allows for the outer skin of the keel to be pierced without water being able to

enter the boat. Figure 1 shows the dimensions of all components in MOOP1 (the second MOOP to be built) and Figure 2 shows the internal electronics and component layout of MOOP1. The rest of this section describes the differences between each variant of the MOOPs and the reasoning behind those differences.

2.1 MOOP0 and MOOP1

The first boat to be constructed was known as MOOP0 and was begun in late 2008. Its hull design differs slightly from the others due to the way it was constructed. A series of 15 cross sections were printed on paper from a design drawing and cut into shape. These were then glued to sections of polystyrene approximately 5 cm

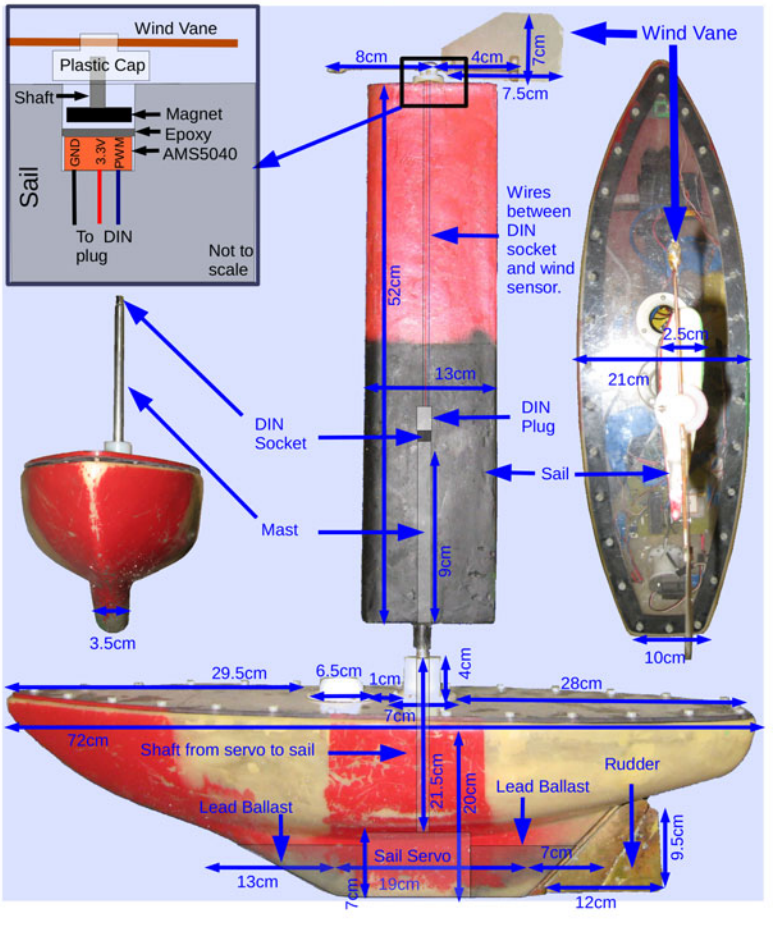


Fig. 1 A diagram showing the dimensions of a MOOP hull and sail

thick and cut using a hot wire cutter to match the shape of the cross section printout. Each of these sections were then joined together to form a polystyrene hull, this was then smoothed off and covered in glass fibre and epoxy resin. The inside of the hull was then hollowed out where required. This resulted in much of the hull still being full of polystyrene, this added buoyancy but restricted the amount of space available inside the hull. The use of relatively coarse polystyrene also left a rough feeling to the entire hull. As shown by the photograph in Figure 3, the deck was split into two sections with a section of glass fibre across the middle between them. The mast was placed in the centre of the boat between the two deck sections. By placing the mast in this location it was hoped that it would reinforce the mast. An aluminium bulkhead was added beneath this middle section to split the inside of the boat into two compartments. Two holes were made near the top of this bulkhead to allow cabling through, but it was hoped that this would prevent water from spreading through the boat in the event of the hull being breached or leaks occurring through the deck. A Honeywell HMC6343 solid state compass, Microchip PIC 18LF4550 microcontroller, 2 line LCD screen for debug messages (a Newhaven NHD 0220JZ FSPG GBW) and Gumstix Connex single board computer were placed in the front compartment and the GPS receiver, batteries, servos, power switch, charging cable and USB cable, reset switch and re-program switch for the PIC were located in the stern compartment. To experiment with weight distribution several small plastic bags full of lead shot were taped inside the hull and moved around as required in order to balance the hull when it was sat in the water. The wing sail featured an attempt to build

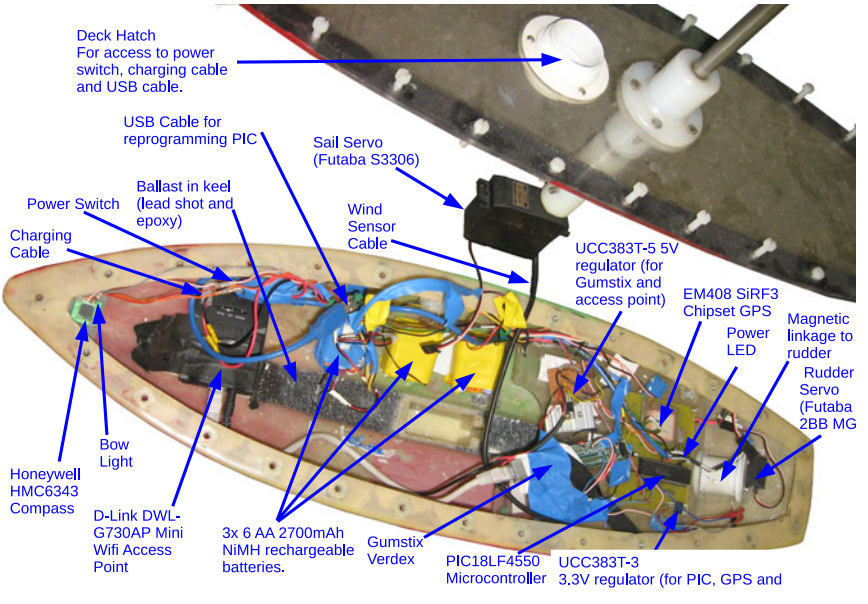


Fig. 2 A digram showing the internal layout of components in MOOP1

an ultrasonic wind sensor using a single 40 kHz transmitter and two receivers. One receiver picked up the signal in the X plane and the other in the Y plane. By amplifying the received signal, combining it with the original signal in a NAND gate and charging a capacitor, this produces a voltage proportional to the time of flight for each sensor. Although this approach worked reliably under laboratory conditions it was exceptionally sensitive to temperature changes and water droplets on the sensors. It is discussed in more detail in [10]. Eventually this sensor was replaced with a continuous rotation potentiometer and later a rotary magnetic encoder. Work on MOOP1 was begun in early 2009 and aimed to produce a simpler, easier to construct hull. A balsa wood plug carved using a similar principle of multiple cross sections that was used in MOOP0's polystyrene core. Using balsa wood allowed a much smoother finish to be created than with the polystyrene. The plug was then covered in glass fibre and epoxy to create a mould (for half the hull) from which other hulls could be made. A photograph of this is shown in Figure 4. Once the two halves of the hull had been made they were glued together with epoxy resin. The deck was cut from a single piece of acrylic plastic. To prevent the deck flexing around the sail plastic blocks were placed around the mast between the sail servo and the deck. A shelf was then created inside the hull where electronics could be placed. The single piece deck required more screws to be undone in order to access the inside of the boat, caused the sail assembly to be lifted off with the deck but made access to the electronics far easier than with MOOP0.

The control system software for MOOP0 and MOOP1 was split into two portions. A low level layer was run on the (PIC18LF4550) microcontroller and interfaced to the servos, compass, wind sensor and GPS. It received commands via a 4800 bps software serial port from the Gumstix Single Board Computer, which was running a Linux based operating system. Through this interface the Gumstix could send target positions for the servos or read data from the compass, GPS or wind sensor. The main control loop would read the compass and set the rudder at a frequency of approximately 8-10 Hz, the wind sensor would be read and the sail set at a frequency of 0.33 Hz. The high frequency response rate for the rudder was required to keep the boat on course and to counter a tendency to turn towards the wind during gusts. Although it would have been possible to implement the control system entirely on the PIC, the Gumstix provided the advantages of easy remote access and logging of data to a filesystem.

A small debugging LCD screen was also connected to the PIC and would report which command had been sent and its response. The Gumstix was also able to send custom messages to this to show when it had finished booting or shutting down. On MOOP0 a Compact Flash 802.11b wireless network card emulated a wireless access point and on MOOP1 (which used a newer Gumstix Verdex and had no compact flash option) an ethernet cable connected the Gumstix to a DLink DWL-G730AP access point. In previous work it had been found that placing an access point on the robot provided a more robust solution than having the robot act as a client to a shore based access point, as doing so meant that the human operator of a laptop was responsible for re-establishing connection rather than relying on this process being automated by the robot. A single UDP broadcast telemetry packet was sent at

a rate of 1 Hz. This packet contained the current actuator positions, GPS location, the location of and heading/distance to the next waypoint, compass heading, wind direction and if the course was directly sailable or not. This data was also logged to a file upon every iteration of the main loop for replaying later on. The data could be received by a remote laptop and displayed upon a moving map application [11]. The use of a UDP broadcast packet allowed telemetry to be received despite high packet loss rates, for multiple users to receive the data and for data to be received by systems which do not even have an IP address on the boat's subnet (as obtaining a DHCP lease from the boat at long distance can be problematic). The use of a Gumstix and wireless network came at quite a high cost to power consumption with the Gumstix and wireless network equipment using between 400 mA and 800 mA at 5 V. This accounts for a significant proportion ofthe robot's power consumption and restricted run times to approximately 4 hours using 18 2700 mAh batteries connected in 3 parallel packs.

During summer 2009 MOOP0 and MOOP1 underwent a significant amount of testing to improve their control systems. In this process several hardware changes were also made. The rudders of both boats were doubled in area, this helped the boats to turn at low speed (usually after a failed attempt to tack) or in strong winds. Two LEDs were placed inside the hull immediately underneath the deck, one at the stern and one at the bow. These helped to locate and determine the direction the boat was facing during darkness or heavy fog. The wind sensors were changed from potentiometers to, more accurate rotary magnetic encoders.

2.2 MOOP2

MOOP2 was constructed in summer 2009 with the aim of sailing across the Atlantic Ocean in the (later cancelled) 2009 Microtransat Challenge. To meet the power budget requirements it was equipped with 10 size F, 1.25 V, 13 Ah batteries connected as 2 parallel packs of 5 batteries. Solar cells were attached to the underside of the front half of the deck, these were expected to supply an average of only 80 mW of power. The transatlantic journey was expected to take approximately 200 days and assuming the batteries started the voyage fully charged and ended it

Fig. 3 MOOP0 sailing in the Atlantic Ocean during the 2009 WRSC in Portugal

Table 1 The specification of MOOP0 and MOOP1

Name	MOOP0 and MOOP1
Date of Construction	Late 2008 (MOOP0) and early 2009 (MOOP1)
Sails	Single Wing Sailboats
Sail Actuator	Servo
Rudder Actuator	Servo on magnetic linkage
Computers	Gumstix and Microchip PIC18LF4550
Wind Sensor	Self made ultrasonic and rotary
Compass	Honeywell HMC6343 I2C solid state tilt compensated compass
GPS	Globalsat SiRF3 EM-408 GPS
Batteries	18 2700 mA h NiMH rechargeable AA batteries
Notes	MOOP0 sailed at WRSC2009, MOOP1 at WRSC2010

almost discharged this gave an average power budget of only 2.68 W h per day. In order to achieve this the entire control system was placed on the PIC microcontroller, eliminating the power consuming Gumstix and wireless network used in MOOP0 and MOOP1. It was intended that the PIC would spend most of the time in a low power sleep mode, waking only every few minutes to read the compass and set the rudder servo. The GPS and wind sensor would be sampled only every few hours and only then would the target course or sail position be changed. The wind sensor was a Furuno Rowind ultrasonic sensor which we had previously used in larger sailing robots, where these had proven themselves to be highly accurate and reliable. In order to track the robot's position a Spot satellite messenger [1] was modified to transmit the boat's position every few hours. This was powered by a separate set of AA lithium batteries so that it would continue to function in the event of the rest of the control system failing. The full specifications of MOOP2 are shown in Table 2 and a photograph of it sailing in the sea off Aberystwyth is shown in Figure 6.

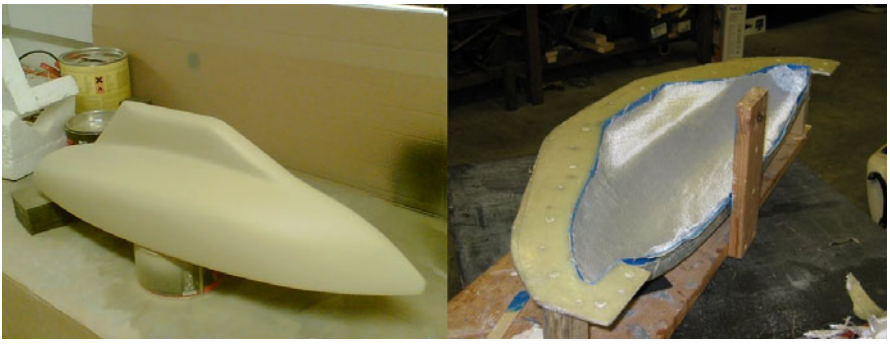


Fig. 4 The plug (left) and the mold (right) for the MOOP hulls

Table 2 The specification of MOOP2

Name	MOOP2
Date of Construction	Summer 2009
Sails	Single Wing Sail
Sail Actuator	Servo
Rudder Actuator	Servo on magnetic linkage
Computers	PIC18LF4550
Wind Sensor	Furuno Rowind
Batteries	10 13 A h NiMH rechargeable size F batteries
Compass	HMC6343
GPS	SiRF3 Globalsat EM-408
Other	Solar Panels, SPOT
Notes	Built to enter the Microtransat Challenge

2.3 Twin Wing Designs

In summer 2009 work began on a twin wing sail MOOP known as MOOP3. Based upon work by Benatar, Qadir, Owen and Baxter [6] using the twin wing sails of a 1.5 m long sailing robot to perform steering instead of the using the rudder this boat was constructed with the idea of further investigating if the rudder could be removed. By turning the rear sail into the wind and stalling it the boat would rotate away from the wind and by turning the front sail into the wind the boat would turn towards the wind. The primary motivation to removing the rudder was to eliminate the complexity of the rudder assembly, which had proved difficult and time consuming

Fig. 5 The rudder assembly, magnetic linkage and servo

Fig. 6 A photograph of MOOP2 sailing in the sea off Aberystwyth

Table 3 The specification of MOOP3

Name	MOOP3
Date of Construction	Summer 2009, added a Gumstix and rotary wind sensor in June 2010
Sails	Twin Wing Sail
Sail Actuator	Modified 360 degree servos
Rudder Actuator	Servo on magnetic linkage
Computers	PIC18LF4550 (Gumstix added in June 2010)
Wind Sensor	Initially no wind sensor, rotary sensor added in June 2010
Batteries	5 13 A h NiMH rechargeable size F batteries
Notes	Competed in the 2010 WRSC

to construct. During this process the possibility of also removing the wind sensor was suggested. As one of the most vulnerable components in the boat this would eliminate a major potential point of failure. By setting the sails for a given point of sail it was hoped that the boat would settle upon a course and that by observing the compass heading and roll of the boat that it would be possible to determine when the boat had settled down. Now that it was known which direction the boat was travelling for a given point of sail the wind direction could be derived. Since originally proposing this method Xiao et al. [14] have devised and demonstrated alternative method based upon a Voronoi diagram, this work suggests that it is quite possible to operate a sailing robot without a wind sensor. It was found that the Futaba S3306 servos used in the previous MOOPs had only about a 200 ° range of turn and that this was insufficient for the sail positions required to sail without a rudder. To overcome this the servos were modified to allow 360 ° rotation, a continuous rotation potentiometer was placed around the shaft and a Polulu Qik 2s12v10 motor controller was used to control the motor from the PIC. In summer 2010 MOOP3 was entered into the World Robotic Sailing Championships and Sailbot competition held in Kingston, Ontario, Canada. Before entering it underwent a few changes, a Gumstix was added to allow it to run the same control system software as MOOP0 and MOOP1 and a wind sensor was added. As there was no wind sensor cabling in the masts/sails, the wind sensor was placed on a pole extending up from the deck hatch. A photograph of MOOP3 can be seen in Figure 7 and its full specifications are shown in Table 3.

2.4 MOOPn

MOOPn was constructed in Spring 2010 and sold to Nottingham University. It featured twin wing sails, a rudder of the same design as MOOP1 and MOOP2, its wind sensor used an AS5040 magnetic encoder (as used by MOOP0 and MOOP1), a Gumstix single board computer and DLink pocket access point were used. The internal layout of the boat was slightly different to all previous boats. All electronics except the rudder servo were attached to a plastic shelf inside the hull, which in turn was connected to the deck. This allowed all electronic components to be lifted out of the

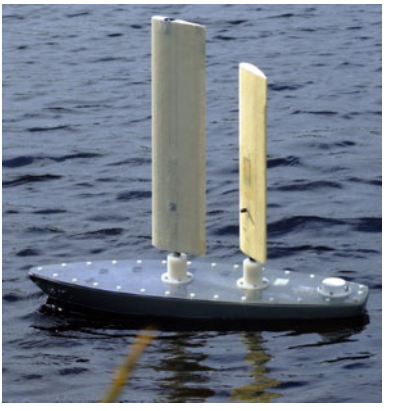


Fig. 7 A photograph of MOOP3 during radio controlled tests

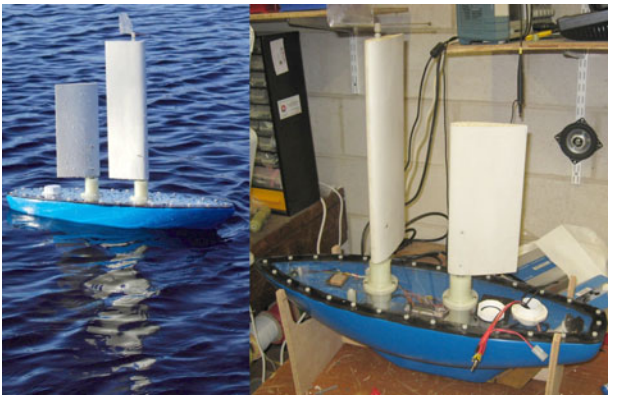


Fig. 8 A photograph of MOOPn both in the laboratory and sailing in Llyn-Yr-Orefa during its acceptance testing

Table 4 The specification of MOOPn

Name	MOOPn
Date of Construction	March 2010
Sails	Twin Wing Sail
Sail Actuator	Modified 360 degree servos
Rudder Actuator	Servo on magnetic linkage
Computers	PIC18LF4550 and Gumstix
Wind Sensor	Rotary wind sensor
Batteries	18 2700 mA h NiMH rechargeable size AA batteries
Notes	Sold to Nottingham University

boat as one unit, which eased diagnostics and repairs. A photograph of MOOPn is shown in Figure 8 and full specifications are shown in Table 4.

3 Experiments and Results

3.1 Lake Tests in Aberystwyth

3.1.1 Power Management Experiments

A series of experiments were carried out using MOOP0 and MOOP1 at a small lake called Llyn-Yr-Oerfa (52.4 degrees North , 3.87 degrees West) approximately 20km East of Aberystwyth. Due to the local topography these were primarily carried out on days with easterly or westerly winds as other wind conditions would lead to rapidly varying wind directions across the lake. The primary aim of these experiments was to develop power management algorithms based upon an abstraction of the mammalian endocrine system, which is responsible for controlling a number of biological processes within the body. These experiments showed that it was possible to use an artificial endocrine system to modulate the magnitude of rudder and sail actuator movements to control power consumption of a sailing robot and that these could be adjusted in response to internal conditions such as battery level or external conditions such as sunlight levels. Further discussion of these results can be found in [12]. These experiments also provide a large dataset on how well the MOOP robots perform. In each experiment run, the boats sailed a beam reach back and forth along a 250 m long (roundtrip) north/south course for between two and four hours. Across a total of 25 experiment runs, 64.65 hours of sailing was undertaken and a total distance of 61.27 km was covered. This leads to an average speed of 0.95 km/h. The mean wind speed (based upon only visual observations) was 8.8 knots (16 km/h), with the minimum being approximately 3.5 knots (6.5 km/h) and the maximum being approximately 16.5 knots (30 km/h). Given the small size of the lake the maximum wave fetch can only be approximately 200 m and this tends to limit wave heights to below 10 cm, but also causes very short wave periods typically less than 1 s. It was originally intended to run these experiments by sailing triangular courses which would include both upwind and downwind sailing instead of just reaching. However, the MOOPs performance in both of these was not sufficient to be able to run a repeatable experiment. When sailing upwind the bow of the boat would point in the correct direction, but the actual direction of travel would be several degrees further off the wind resulting in little progress being made. Accidental tacks due to small wind shifts or waves were not uncommon and these could easily undo the last 10 minutes of progress. During one (of the few successful) upwind sailing test it took MOOP1 42 minutes to sail 65 m upwind, the return journey took only 5 minutes. On downwind legs frequent jibes were experienced. Attempts were made to perform downwind tacking to control this, although to completely prevent unintended jibing the boat needed to sail as much as 45 ° of the wind.

3.1.2 Twin Wing Sail Experiments

Initial experiments with MOOP3 used a radio control system to steer the boat and test its ability to sail a variety of courses. No rudder was used during these experiments. There were two aims to this experiment, first to determine if the boat would be stable on all points of sail without a rudder and secondly to test if by setting the sails correctly the boat would settle onto a given point of sail from an arbitrary starting condition. If the latter were true then it would allow for sailing without a wind sensor as the wind direction could be derived simply by setting the sails, waiting for a few minutes and establishing the direction of travel. It was discovered during this experiment that it was possible to steer the boat onto any point of sail but that it was difficult to remain on course when sailing downwind. Typically after only a few seconds of downwind sailing the boat would turn towards the wind. It was also found that the boat would not always settle on a predictable course for a given set of sail positions and that there were often multiple possible courses (including sailing backwards) that could be sailed for any given position. A video of this experiment is available from [2].

MOOPn was developed with the intention of using a rudder and twin wing configuration together. Its control system was implemented to use the sails to assist in the steering process. Only a total of 11 unique positions were allowed for each sail, this design had been inherited from MOOP0 and MOOP1 which suffered considerably from poor repeatability of their sail servos and 11 positions had been the maximum that could be achieved in a consistently repeatable manner. The sail assisting process used the rear sail only and would move it by a maximum of one position. If the boat was heading too close to the wind then the sail would be let out one position and if it was too far from the wind the sail would be pulled in by one position. The sails would also be placed into a "goose swing" position (one sail on each side) when the wind was coming from a 20 ° wide sector immediately behind the boat. This avoided downwind stability problems previously found in MOOP3. MOOPn was taken for an acceptance test sail on Llyn-Yr-Orefa and successfully completed a triangular course. Although a single wing MOOP was not used for comparison at the time, the performance of MOOPn was a noticeable improvement. There were none of the violent jibes seen by MOOP0 and MOOP1 when sailing downwind. Upwind performance also appeared more stable with no accidental tacks being noted. A video from this test is available from [3].

3.2 Sea Tests

In addition to lake tests, several sea (and large lake) tests have also taken place with various MOOPs.

3.3 WRSC 2009

The first sea test took place in the Atlantic off the coast of Matosinhos, Portugal during the World Robotic Sailing Championships in July 2009. A triangular course

of approximately 1.2 km in length was set. The first two portions of this were on a broad reach and the final was straight upwind. MOOP0 was used and successfully sailed the first two portions of the course but then failed to make the turn onto the third. This was believed to be due to the tidal current which was moving in the same direction as the wind. The boat was then carried upwind and allowed to sail back to the second waypoint with the wind now straight behind the boat. It made progress towards this waypoint, but jibed almost constantly in the process. This test took place during calm weather (Beaufort force 1 or 2) and so cannot be considered to have covered conditions representative of those on the high seas.

3.4 Aberystwyth Sea Test

MOOP2 had been constructed with the intention of cross the Atlantic ocean during the (later cancelled) 2009 Microtransat Challenge. A short sea trial was undertaken in August 2009 in the waters off Aberystwyth. A waypoint was set approximately 200 m offshore and a second waypoint was set onshore, with the intention of having the boat sail itself ashore (in order to test the concept of setting the final waypoint ashore, to have the robot beach itself). The robot was sailed some distance towards its first waypoint but encountered some difficulties. On several occasions the robot span around in circles and due to the weight of the wind sensor it constantly swayed from side to side after each wave passed. Eventually the chase boat decided that there may have been some software issues and to abort the attempt and advance the boat to its next waypoint. The robot was able to successfully sail back towards the beach, sailed through the (relatively small - 0.5 m) surf and onto the beach. This process was repeated three times to test repeatability and if the boat would actually become beached or if it would be swept away by the next wave. After the third attempt several dents and holes were noticed in the hull and further tests were aborted. The level of damage which had occurred in low levels of surf onto a mixed pebble and sand beach suggested that, attempting to autonomously beach a MOOP will almost always be damaging to the hull and in more severe situations would probably result in serious damage. The swaying caused by the wind sensor is a serious issue and suggests that the wind sensor weighed too much (approximately 300 g), therefore it will not be possible to use a Furuno rowind sensor on a MOOP.

3.5 WRSC 2010

MOOP1 and MOOP3 were entered into the 2010 World Robotic Sailing Championships in Kingston, Ontario, Canada. They both struggled against a current of approximately one knot crossing the racing area. Both boats were able to successfully sail with the current but never completed an entire course due to the current. It was interesting to note that the 53 cm MicroMagic boats from the University of Lübeck [5, 4] were able to complete the same course. This suggests it is specific design elements of the MOOP rather than simply the lack of hull speed in any small boat causing these problems. These boats featured much narrower keel designs than the MOOPs and therefore are likely to have encountered far less drag than a MOOP hull.

4 Conclusions and Future Work

Overall the MOOPs have proven themselves to be a cheap, flexible and reliable platform for developing sailing robot control systems. However, they have struggled with sailing upwind and against currents. The twin wing sail designs have shown themselves to be more stable on downwind courses and offer the potential of rudderless sailing or, at the very least that the rudder and sail can be used cooperatively. Despite the name "Miniature Ocean Observation Platform" it remains unlikely that a MOOP is likely to be able to complete any serious ocean sailing unless it is sailing downwind and with the currents. This might for example, be achievable between the Canary Islands and the Caribbean, but it is unlikely that a MOOP will be able to sail from the UK or Ireland to the Canary Islands which rules out an entry into the Microtransat Challenge following its 2010/2011 rules.

When compared with other small boats such as the MicroMagic boats used by the University of Lübeck [7, 4] or Breizh Spirit built by ENSTA-Bretange [13] upwind performance of the MOOPs is much poorer. This is believed to be due to the drag caused by the wide keel design. If serious ocean sailing against tides and currents is to be achieved then a redesign of the hull shape may be required.

Further sea trials of the MOOPs are required. Reductions in power consumption are also required for prolonged sailing. One possibility is to use a self correcting wing sail similar to that used by Elkaim and Boyce [8], where a small rudder on the wing sail is used to control its angle, once the rudder is set the sail will maintain its position with respect to the wind without the need for an active control system. To test the longevity of components and ability to sail at sea one proposed experiment is to drop an MOOP without a control system or only a very simple heading holding system into the sea and monitor its progress via a satellite tracker.

Acknowledgements. The authors would like to thank Barry Thomas and Tom Blanchard for their contributions towards designing, constructing, testing and repairing the MOOPs.

References

1. Spot satellite messenger, http://www.findmespot.com (accessed June 10, 2011)
2. Dual wing sailed moop sailing robot (2009), http://www.youtube.com/watch?v=X9Hj3jcL5vk (accessed June 10, 2011)
3. Twin wing miniature sailing robot (2010), http://www.youtube.com/watch?v=w8mFVsT23Tw (accessed June 10, 2011)
4. Ammann, N., Hartmann, F., Jauer, P., Bruder, R., Schlaefer, A.: Design for a robotic sailboat for wrsc/sailbot. In: Proceedings of the 3rd International Robotic Sailing Conference, Kingston, Ontario, Canada, pp. 41–43 (June 2010)
5. Ammann, N., Biemann, R., Hartmann, F., Hauft, C., Heinecke, I., Jauer, P., Kruger, J., Meyer, T., Bruder, R., Schlaefer, A.: Towards autonomous one-design sailboat racing: navigation, communication and collision avoidance. In: Proceedings of the 3rd International Robotic Sailing Conference, Kingston, Ontario, Canada, pp. 44–48 (June 2010)

6. Benatar, N., Qadir, O., Owen, J., Baxter, P., Neal, M.: P-controller as an expert system for manoeuvring rudderless sail boats. In: UK Workshop on Computational Intelligence (UKCI 2009), September 7-9, University of Nottingham (2009)
7. Bruder, R., Schlaefer, A., Stender, B.: Model sailboats as a testbed for artificial intelligence methods. In: Proceedings of the 2nd International Robotic Sailing Conference, pp. 37–42 (2009)
8. Elkaim, G., Boyce, C.: An energy scavenging autonomous surface vehicle for littoral surveillance. In: Proceedings of ION Global Navigation Satellite Systems Conference (2008)
9. Neal, M.: A hardware proof of concept of a sailing robot for ocean observation. IEEE Transactions on Oceanic Engineering 31(2), 462–469 (2006)
10. Neal, M., Sauze, C., Thomas, B., Alves, J.C.: Technologies for autonomous sailing: Wings and wind sensors. In: Proceedings of the 2nd International Robotic Sailing Conference, Matosinhos, Portugal, July 6-12, pp. 23–30 (2009)
11. Sauze, C.: Sailing robot route planner. Microtransat Challenge Sourceforge Project (2009), http://microtransat.svn.sourceforge.net/viewvc/ansat/route_planner (accessed June 10, 2011)
12. Sauzé, C.: A neuro-endocrine inspired approach to power management in sailing robots. Ph.D. thesis, University of Wales, Aberystwyth (2010)
13. Sliwka, J., Reilhac, P., Leloup, R., Crepier, P., Malet, H.D., Sittaramane, P., Bars, F.L., Roncin, K., Aizier, B., Jaulin, L.: Autonomous robotic boat of ensieta. In: Proceedings of the 2nd International Robotic Sailing Conference, Matosinhos, Portugal, July 6-12, pp. 1–8 (2009)
14. Xiao, K., Sliwka, J., L. Jaulin, K.R.: A wind-independent control strategy for autonomous sailboats based on voronoi diagram (2009)

Breizh Spirit, a Reliable Boat for Crossing the Atlantic Ocean

Richard Leloup, Frédéric Le Pivert, Sébastien Thomas, Gabriel Bouvart, Nicolas Douale, Henry De Malet, Laurent Vienney, Yvon Gallou, and Kostia Roncin

Abstract. To meet the Microtransat challenge, ENSTA Bretagne chose to realize several sailing robots. The first, having served as a test platform, allowed us to develop two new boats, one for research and one for the Atlantic crossing. The different tests in the bay of Brest allowed us to improve the reliability of systems on our sailboat. Various studies have been conducted to improve reliability of sailboats both mechanically and electronically. This tests allowed us to test different concepts on the three Breizh Spirit boats. The results are very positive and we can now say that we have a boat able to resist to strong storms, to follow a predifined route, to supply its own energy and to navigate in sea waves. From the experience acquired from Breizh Spirit 1, we hope we will be able to cross the Atlantic Ocean.

1 Introduction

As part of the international Microtransat challenge between different universities and scientific schools, ENSTA Bretagne decided to develop several sailing robots. In 2009, ENSTA Bretagne developped a 1.3m sail-boat (Fig. 1) named Breizh Spirit 1. The boat meets several of the required criteria [3], i.e. a sailing boat able to follow a predefined route in fully energetic and decisional autonomy. It was capable of crossing the Bay of Brest in September 2010 as a validation test. Nevertheless, this boat was largely destroyed during a test between Brest and Morgat. To establish models of behaviour at sea for crossing the Atlantic Ocean, ENSTA Bretagne

Richard Leloup · Frédéric Le Pivert · Sébastien Thomas · Gabriel Bouvart · Nicolas Douale · Henry De Malet · Laurent Vienney · Yvon Gallou · Kostia Roncin
École Nationale Supérieure des Techniques Avancées (ENSTA) Bretagne, 2 rue François Verny, 29806 BREST cedex 9, France
e-mail: richard.leloup@ensta-bretagne.fr

Frédéric Le Pivert
Université de Bretagne Occidentale (UBO), 6 Avenue Le Gorgeu, BREST, France
e-mail: frederic.lepivert@etudiant.univ-brest.fr

Fig. 1 Breizh Spirit 1 on the Ty Colo Lake

decided to build a second boat named Breizh Spirit 2 which is fully instrumented and drawn from studies in CAD (Computer-Aided Design). Finally, in view of participating in the Microtransat challenge in November 2010, the students in Naval Architecture (ANO) and Mechanical Engineering decided to start the design and construction of Breizh Spirit 3 which is 1,4 m long, and designed to be able to cross the Atlantic Ocean autonomously, based on the experience of the previous two sailboats.

The main objective for us, as we will present in this paper is to improve the reliability of Breizh Spirit for the Microtransat challenge. The design of the last two boats is the result of several studies based on the experiences of Breizh Spirit 1 and other boats. In the first part we develop some points which were particularly studied. Moreover, electronics were also completely rebuilt to improve reliability. Nevertheless Breizh Spirit 2, beyond the crossing of the Atlantic, is also used for research programs to study the behaviour of the boat at sea. That is what we will develop in the third part. Finally, we will present the results of our work at different validation tests.

Table 1 Boats Characteristics

-	unit	Breizh Spirit 1	Breizh Spirit 2	Breizh Spirit 3
LOA	m	1.5	2.3	1.7
LWL	m	1.3	2	1.4
BWL	m	0.35	0.8	0.45
T	m	0.8	0.8	0.8
SA	m	0.8575	2	0.75
Disp.	kg	13	55	13
Cb	-	-	0.6	-

Fig. 2 servo-motors protection system

In the Table 1,we characterize our boats by Length Overall (LOA), Waterline Length (LWL), Displacement (Disp.), Waterline Beam (BWL), Draft (T), Sail Area (SA) and Block coefficient (Cb).

2 Development of a Platform Adapted for Crossing the Atlantic Ocean

2.1 Design of a Transatlantic Boat

After the damages caused during the endurance test between Brest and Morgat, we chose to draw conclusions from Breizh Spirit 1 to finalize the construction of Breizh Sprit 3. With regard to the first boat we realised that the weight of the vessel had been underestimated, the reserve buoyancy became insufficient. That is why the first sailing boat plans, [5], and expanded the hull and increased its height while keeping the same form which showed a very good behaviour in manoeuvrability and seakeeping. As explained in [5], we chose to inspire the forms of an Open 60 boat from the class IMOCA which is accustomed to facing the Atlantic Ocean at various regatta.

Breizh Spirit 1 had also shown that it is difficult to have a sail boat completely waterproof. Various technologies exist to solve this problem. For example we can use an automatic bilge pump to drain water [4]. However, this solution is very energy intensive. We can also choose to seal as perfect as possible, but this solution is very expensive and not always very reliable [2,7]. That is why we chose to make a completely unsinkable sailboat building it from blocks of closed cell foam. So, even if the boat suffer any collision damaging the hull, as we saw with Breizh Spirit 1, there is no risk of sinking because the rest of the hull will keep its own reserve of buoyancy. However, the integration of electronics and actuators requires waterproof areas. That is why we chose to make a boat with three zones with three different levels of sealing:

- The first level is the area behind the cockpit which is semi-sealed, protected by the solar panels and it contains all components as we can see in Fig 4. ;
- In this area, the movable portion of the actuators is placed in a watertight compartment almost like the one for Breizh Spirit 1, in which water can enter only by the outputs of the sheets. ;
- The servo motors and any part of electronics are placed in a fully watertight compartment as we can see on Fig. 2 And Fig. 4. For this, we chose to use IP 67 certified boxes, which are fully waterproof and suitable for use in very hostile marine environments.

The Brest-Morgat test also demonstrated that all that exceed outside of the hull may be torn by waves or collisions. Indeed, on the Breizh Spirit 1, we chose to put the actuators and the sheets outside. Both have been uprooted and lost during the Brest-Morgat test. Therefore, on Breizh Spirit 3, we chose to protect these elements. As a matter of fact, the box containing the actuators is placed under the bridge, just behind the stern, to be fully protected from waves and water infiltrations. The circuit of sheet of Breizh Spirit 1 which was around the boat had shown good reliability, particularly because the sheets didn't bring water inside the hull. We have therefore chosen to design a similar system. Moreover, the entire circuit is placed in machined channels in the hull that allow us to protect the whole circuit with solar panels (Fig 4 and Fig 5). The rudders are controlled in the same manner which limits water infiltration especially in the box containing the actuators. Moreover, the fact of separating the electronic actuators ensures a perfect seal for the battery and electric circuits.

2.2 Use of a Simulator to Design the Rudders

To illustrate our strategy for the choice of the different components, we present in this chapter the entire strategy for choosing the motors for the rudders of Breizh Spirit 2. The other components and architectural choices have been taken in the same manner for Breizh Spirit 2 and Breizh Spirit 3.

One of the most important points for such a small craft is the energy balance, because there is a very small area where to lay solar panels. The question of energy consumption is more important for the rudder than for the sail. You can keep the sail

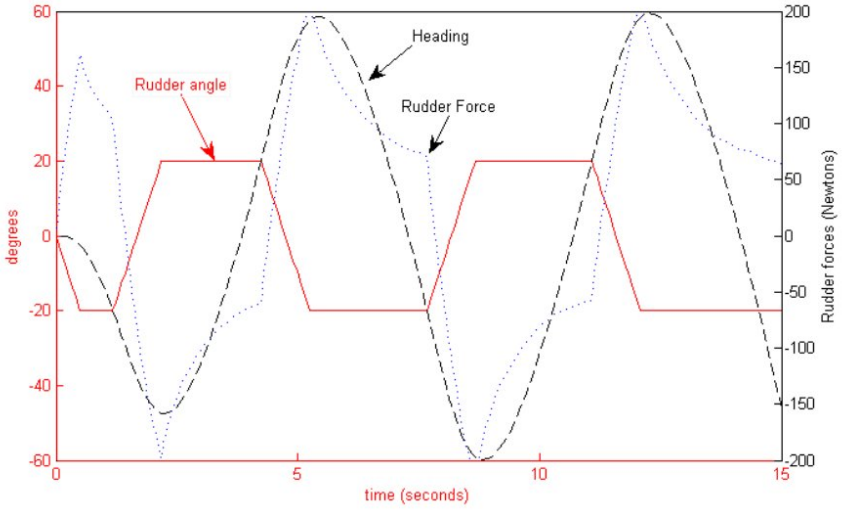


Fig. 3 Evaluation of the force acting on the rudder by manoeuvring Zig-Zag simulation

fixed for a quite long time while you must control the rudder quasi-continuously. Thus the question of the design of the entire steering gear is crucial. We decided to build a 6 degrees of freedom simulator to evaluate the forces acting on the rudder of Breizh Spirit 2 in order to choose an appropriate actuator. The principle of the simulator is thoroughly described in [1]. For the current study we do not need such a precise hydrodynamic model. Thus, we use the very simple and classical one by [9]. For the forces on the rudders and the keel we use the model derived from air-foil theory proposed by [10].

The dimensioning case is a boat speed of 10 knots and the rudder moving alternatively from starboard to port with 20 degrees in magnitude. The time to move the rudder from starboard to port is about one second. The rudder law is driven accordingly to the zigzag test ITTC standards. Fig. 3 shows that for such a configuration the force acting on the rudder reaches 200 N.

2.3 Choosing the Rudder Motor

After a study to determine the necessary torque for the rudder motor, we selected the following actuators:

- servo motor Hitec HS 805 BB +;
- servo motor Dynamixel RX 28;
- servo motor Futaba Robbe BLS 151;

Table 2 Comparison Table

Servo motor model Criteria	HS 805 BB+ Hitec	RX 28 Dynamixel	BLS 151 Futaba Robbe
durability	More than 100 hours	More than 100 hours	More than 1000 hours
Waterproofness	Designed for wet environment	Designed for dry environment	Designed for wet environment
Nominal torque	19.8-24.7 kg.cm	28.3-33.7 kg.cm	9.6 kg.cm
Maximal torque	79.8 kg.cm	113.2kg.cm	14.4kg.cm
Input voltage	4.8-6V	12-16V	4.8V

The comparison table shows various criteria including:

- Durability which is an important factor when considering the crossing of the Atlantic. Longevity depends on the technology used. Indeed, brushless motors (like RX 28 Dynamixel) have a far superior life in comparison to current engines. However, considering a voyage across the Atlantic for about five months, the rudders being controlled only a tenth time to save energy, we can expect that we will use the motors for a period between 300 and 400 hours.
- Waterproofness must be taken into account on a sailboat because it would be faced with a marine environment during the Atlantic crossing. The lack of waterproofness may force us to realize a complex system to prevent water from entering the motor.
- To calculate the torque, we need to determine the movement of the cables which control the rudder during the rotation for an angle of 35 degree and the force on the rudders. The calculated nominal torque taken into account for the rudder is then 127.8 N.cm i.e. 12.78 Kg.cm
- The DC motors having no common current limiter, can withstand peaks equal to 5 times their rated torque. However, the peaks create an overheating of the motor. Brushless motors with a current regulating canŠt withstand overload greater than 50 of their rated torque. According to calculations, the maximum torque supported by the motor is 504.9 N.cm id 50.49 Kg.cm
- the Input voltage gives us information on the consumption of the engine.

The Table 2 clearly shows that, despite their durability, the BLS151 motors can't bear the torque generated by the rudders. The RX 28 has the disadvantage of not being designed for marine environments and therefore requires to be sealed. Moreover it is difficult to use because of its high consumption. Therefore we chose the servo motor HSB 805 BB + from Hitec which seems to be the best for Breizh Spirit 2.

2.4 Construction of the Hull from a Digital Model

For Breizh Spirit 2 and Breizh Spirit 3, we decided to make a digital model of the boat and its components to facilitate the design of elements of the hull. In this section, we will focus primarily on the achieving of Breizh Spirit 3.

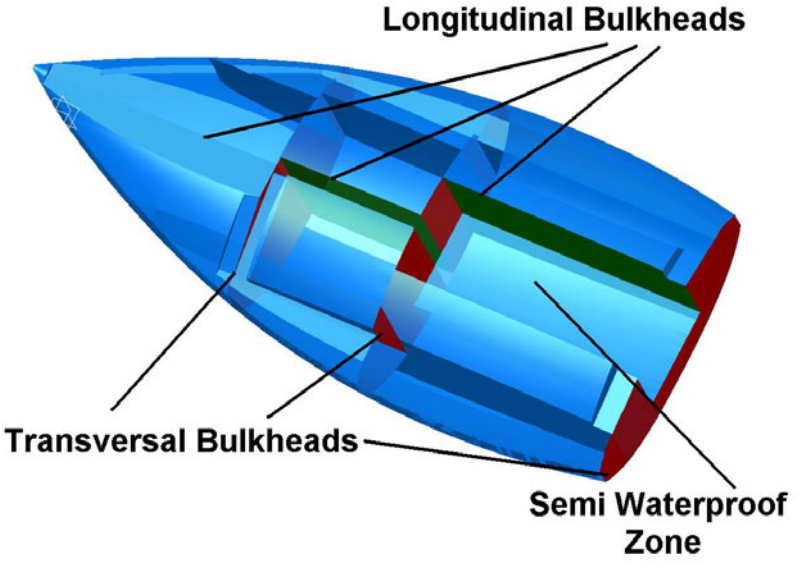


Fig. 4 Digital model of the hull

As we have seen before, using foam blocks has an obvious advantage in terms of unsinkability. The second advantage is that we do not have to make a mold as was done for Breizh Spirit 1. Using the digital model allowed us to obtain a ŞperfectŤ hull shape. However, the choice to build this boat from foam blocks machined with a milling machine required us to fully draw each block. We used the software developed by Dassault System, Catia, which allows us to fully define the geometry of the hull form from a surface recovered from the naval architecture software DelftShip. Once the hull is designed, Catia generates all the programs that are then given to the milling machine for the machining of the foam. The definition of the blocks has been conditioned by different parameters:

- The movements and courses of the milling machine.
- The position of the partitions.
- The integration of components.
- The thickness of the foam raw.

Indeed, because the movement of the machine is limited to 0.5m in length and width, and 0.3m in height, we were forced to carry out at least 6 blocks being given the dimensions of our hull. Otherwise one of the major points during the design of a boat is the consideration of the bending moment and the torsion moment due to the keel and shrouds. That's why we built the hull with longitudinal divisions (in green

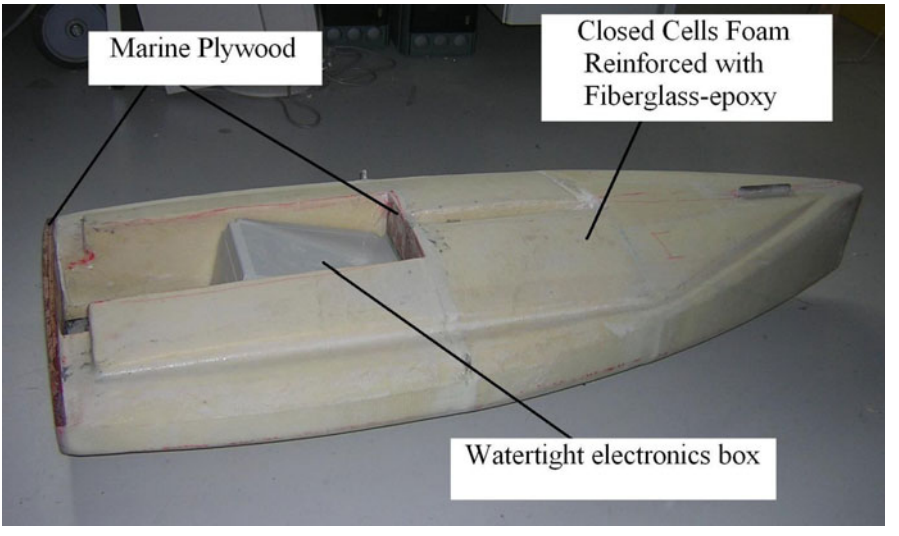


Fig. 5 "Breizh Spirit 3"'s hull

on the Fig 4) to withstand the bending moment due to wave action and tension in the rig. The transverse bulkheads (in red on the Fig 4) withstand the torque moment.

Finally, we got a shell constructed from 13 blocks machined with the milling machine. The cutting of a block of the hull was made from a rectangle of foam. Once the blocks were cut, they were assembled to obtain a stiffened shell with walls made from fibreglass impregnated with epoxy. The boat was finally covered with an external skin using a vacuum pump to impregn fiber with epoxy to creat a composite as we can see in Fig 5.

3 Development of Reliable and Robust Electronics

3.1 Reliable Electronics

To optimize power consumption on-board the electronics has been developed based on PIC microprocessor (PIC 18F2550 to have enough memory and calculation power).The sensors required to control the boat are chosen for their performance and low consumption:

- Wind sensor: Ultrasonic anemometer CV7 (LCF Capteurs, FRANCE)
- GPS: FV-M8 (SANAV, San Jose Technology, Inc., USA)
- Magnetic Compass:HMC6343 (Honeywell International, USA)

The servo-motors (Fig. 2) are powered by a circuit different from the microprocessor (on the centre of Fig. 6). The energy supply can be put on stand-by by the microprocessor in order to limit losses (load current supply and servo motors for

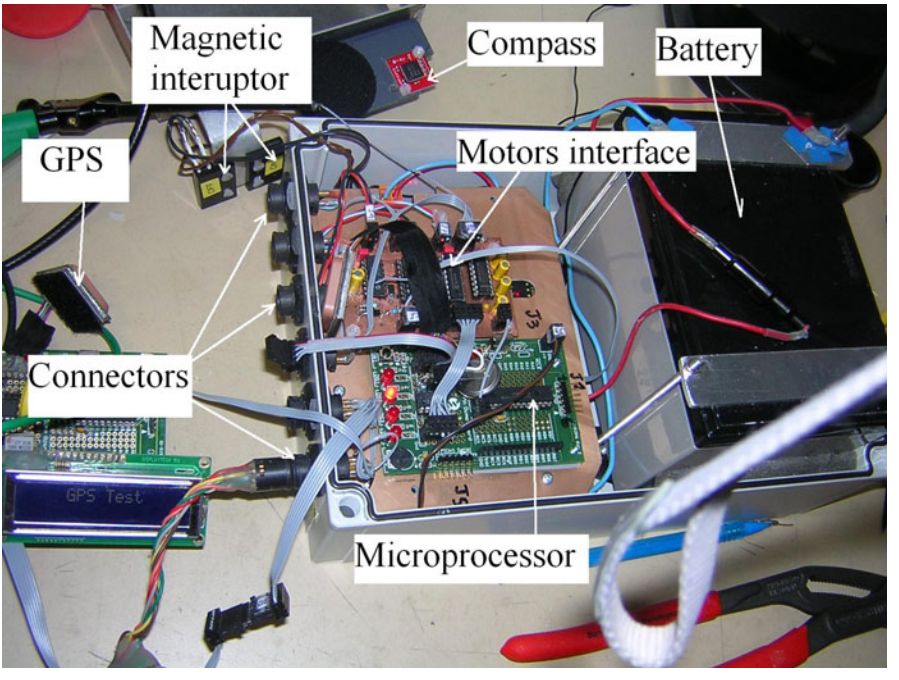


Fig. 6 ńăBreizh Spirit 3ăż's new electronics

the moments). All power supplies loaded (conversion of 12 V 5V, 3.3V, etc.) are selected for their efficiency (0.93). This energy management can lower the average consumption of between 100 and 150 mA. The embedded energy is stored in a battery of 12 Volts 12 Ah (on the right of Fig. 6). Conventional lead acid battery technology was chosen because of its reliability. Such a battery without electrolyte fluid is sufficient for a journey expected to last six months.

This system can supply the energy for 3 days minimum. The recharge of the battery is provided by solar panels that can charge up to 500 mA. These panels can charge the battery during the night operation with a slight margin for the days of low sunlight. For the proposed crossing of the Atlantic (tropical road), such days should be rare. However, in the case of a loss of battery power, a charge controller can turn off the electronics and actuators power. A memory system allows the boat to restart when the battery voltage reachs sufficient capacity to ensure proper operation. During these moments without power, the boat will behave like a drifting raft.

GPS position sent by the SPOT system has, for security reasons, a separate supply by a lithium battery to ensure the proper operation of sending the position three times per day for 6 months. For this, the SPOT casing has been modified to a microprocessor to wake him up only when it must send messages.

The entire equipment is housed in a waterproof case. Links to other elements (solar panels, sensors, microprocessor and memory programming way points) pass

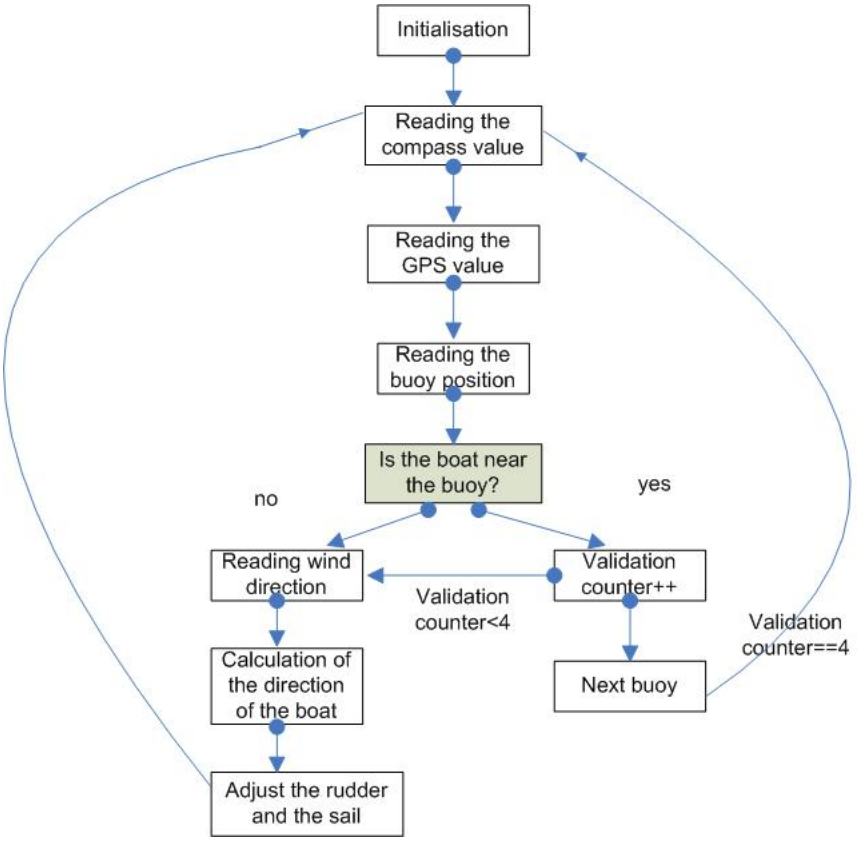


Fig. 7 Navigation algorithm

through a series of waterproof connectors. To ensure a complete waterproofness, a magnetic control system replaces a possible waterproof switch which isn't reliable.

3.2 Hardware and Software Organisation: The Algorithm

As we said before, the boat is using three sensors: a compass, a GPS and an anemometer. With the compass we get the direction of the boat, with the GPS the position. To be sure there is no error, the program make the boat head to the next point after checking four times it is near the buoy as we can see in Fig. 7. The anemometer gives the wind direction to the boat. In order to be not too sensitive to the variation of wind direction, we prefer checking this direction from time to time. We do not follow the wind variation and therefore we choose a minimum angle between wind and boat direction equal to 50 degrees.This angle has been tested and we are sure that the boat may not stop. We also do not adjust the rudders exactly for the precise direction, we prest five discrete rudder angles chosen with four filter

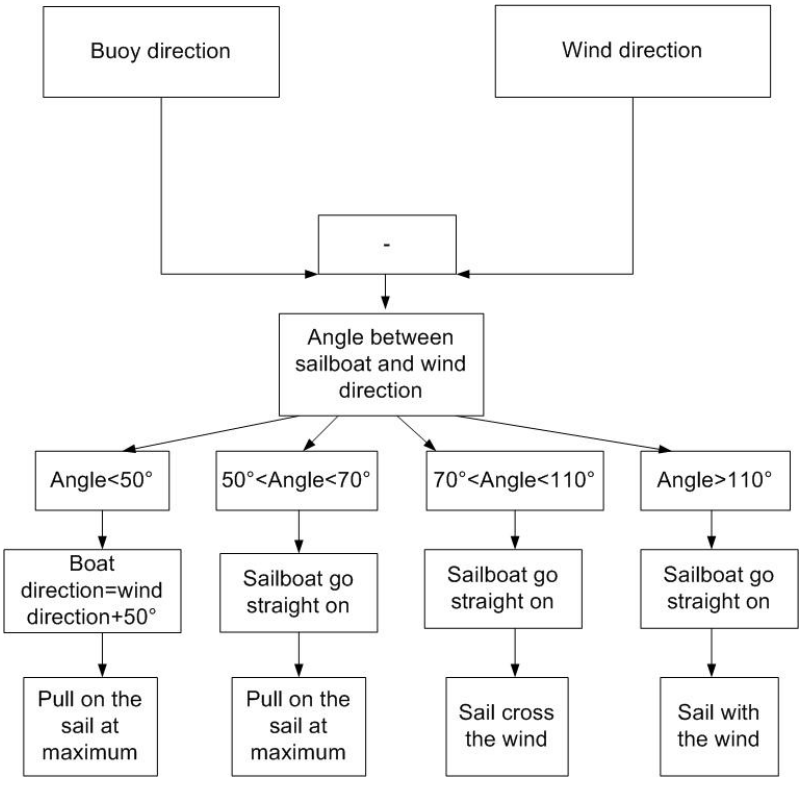


Fig. 8 Direction algorithm

angles: if the angle between buoy and sailboat direction is between 0 and 13 degrees, we do not adjust the rudder, if the angle is between 13ř and 28ř, we turn the rudder of 10 degrees, between 28 and 55 degrees, we turn of 20 degrees, between 55 and 80 degrees, we turn of 30 degrees and finally if the angle is bigger than 80 degrees, we turn the rudder of 45 degrees. We prest the same strategy for the sails angle as shown in Fig. 8.

4 Development of Research Programs around Breizh Spirit

A research program is in current development to provide experimental data to validate the seakeeping codes that are realised at LBMS/ENSTA Bretagne.

This program consists of realising a measurement system for recording the movement of a boat at swell. The interest of using a sailing robot is first off all its small size. Thus sailing in a usual seastate condition gives situations where the waves become big compared to the boat dimensions.

The second interest is the control we have on each actuator of the boat. This is very important, because with classical sailing boat tests there are big uncertainties concerning the crew behaviour, the positioning and the trimming of sails [8]. Of course it is possible to get information from video recording. But it complicates and delays the analysis of the experimental data recording. Sailing robots have the advantage that all parameters can be easily known at each time step of the experiment.

5 Tests and Results

The very first tests of Breizh Spirit 1 were conducted during the WRSC 2009 in Porto. Obviously, they were not successful. From discussions with the other

Fig. 9 Brest Harbour crossing. The yellow points correspond to the waypoints. Breizh Spirit 1 sailed 6.5 nm at average speed of 3 knots.

participants it was clear that the development of such a project should lean on three tools:

- a light prototype, easy to carry and implement
- a simulator, useful to detect bugs and to avoid spending time on water.
- a monitoring system to get information from the tests and understand the boat behaviour.

During the following weeks a monitoring system was implemented. The first successful tests were carried on the small lake Ty Colo near Brest. Basically the boat managed to turn around 3 way-points. Then it was decided to attempt the crossing of the Brest Harbour from Cosquer to Lanveoc in early September. The wind was north-east 15 to 20 knots during the main part of the crossing and then dropped to less than ten knots and turned east to north-east at the end. The boat was totally autonomous until it reached the Lanveoc shore. Then we took the control to end the course and to pass the Pen ar Vir point just in front of the French Naval academy. Breizh Spirit 1 did 6.5 nautical miles at an average speed of 3 knots, and it reached several times a maximum of 5.6 knots.

Fig. 10 Track of the Brest-Morgat test. On the center of the figure, we can see that Breizh Spirit 1 couldn't tack upwind on 30 knots wind speed.

The following year was dedicated to enhance reliability and to develop a system validation and verification procedures for each sensor and actuator. It remained to test the behaviour of the vessel at strong sea state. The vicinity of the Ushant Island (Ouessant in French) often offers extremely hard sea conditions, especially when the wind blows against the current. Thus it is an excellent playground for testing seakeeping ability and endurance of our platform. Two attempts were conducted and permitted to detect some weakness in our electronic and mechanical systems.

For the second attempt around Us Island, Breizh Spirit 1 travelled 12 nautical miles upwind, tacking in waves up to 2 to 3m. Its behaviour was very satisfactory beyond our own hopes. We considered then that only the endurance function remained to validate for crossing the Atlantic Ocean.

For the third attempt the meteorological forecast was so uncertain that we chose only to go round the peninsula of Crozon from Brest to Morgat. Fig. 10 shows the track recorded before the crash of Breizh Spirit 1. One can see on course that there is a strong drift of the boat between the way-point. This drift is due to a bad setting of the rudder servo. This servo was changed for a better model just before the test and we just forgot to do new controls. As a consequence, for each way-point, the boat had to tack and go upwind to reach its way-point. The validation circle is about 400m diameter. When it was about to reach the Swensea Vale Way-point the wind suddenly strengthened up to 30 knots. As we can see the boat was no longer able to tack into the wind. Then it was tacking upwind for approximately 10 hours before it crashed on the rocks. Due to problems of accessibility, the boat was left there for three days. Two storms destroyed the entire rig and appendages, but the electronic system was intact despite some water ingression.

6 Conclusion

As a conclusion, we can say that the experience gained from Breizh Spirit 1 has enabled us to design a boat, Breizh Spirit 3, much more suited to cross the Atlantic. Indeed, we found some errors in the design of the Breizh Spirit 1. The electronics has

Table 3 Progress Table of Breizh Spirit

Test Platform (BS1)	Validated
Electronic architecture for BS1 and BS3	Validated
Autonomous navigation of 12 miles with BS1	Validated
Energetical design for 6 months of BS3	Validated
Mechanical design for Crossing the Atlantic of BS1 and BS3	Validated
Control of component's reliability with BS1	Validated
Setting Controls for BS3	In progress
Endurance test with BS3	In progress
Construction of BS3	In progress

been entirely rebuilt to ensure its durability. Now, the Microtransat project arouses a lot of interest from industry and research. Therefore, we will continue to develop various research programs on Breizh Spirit 2. The eventual aim being to cross the Atlantic, some tests will still be made with Breizh Spirit 3, drawing conclusions from the misadventures of Breizh Spirit 1. Indeed, despite of all the work, as can be seen from the Table 3, it remains to test the endurance of the boat but also its behaviour in wind less than 10 knots and more than 30 knots upwind. With these various experiences, we hope to be able to compete in September 2011 and meet the Microtransat challenge .

Acknowledgements. Thanks to DCNS, a company specialized in naval shipbuilding, to support the project Breizh Spirit. Thanks to all who contributed to the realisation of the Breizh Spirit Project. Especially Michel Jaffres and Bruno Aizier.

References

1. Roncin, K., Kobus, J.M.: Dynamic simulation of two sailing boats in match racing. Sports Engineering 7(3), 139–152 (2004)
2. Briere, Y.: IBOAT: An autonomous robot for long-term offshore operation. In: Proceedings of MELECON 2008. The 14th IEEE Mediterranean Electrotechnical Conference, May 5-7 (2008)
3. Briere, Y., Bastianelli, F., Gagneul, M., Cormerais, P.: Microtransat challenge. In: CETSIS 2005, Nancy, France, vol. 5(2) (2005), doi:10.1051/j3ea:2006031
4. Stelzer, R., Jafarmadar, K.: Communication architecture for autonomous sailboats. In: International Robotic Sailing Conference (IRSC), Porto, Portugal (July 2009)
5. Sliwka, J., Reilhac, P., Leloup, R., Crepier, P., Malet, H.D., Sittaramane, P., Bars, F.L., Roncin, K., Aizier, B., Jaulin, L.: Autonomous robotic boat of ensieta. In: 2nd International Robotic Sailing Conference, Matosinhos, Portugal (2009)
6. Thomas, G.A., Harris, D., d'Armancourt, Y., Larkins, I.: The Performance And Controllability Of Yachts Sailing Downwind In Waves. In: Proceedings of the 2nd High Performance Yacht Design Conference, Auckland, New Zealand, p. 8 (Ferburary 2006)
7. Miller, P., Brooks, O., Hamlet, M.: Development of the USNA SailBots (ASV) (2009)
8. Roncin, K., Kobus, J.-M., Iachkine, P., Barré, S.: Mé,thodologie pour la validation du simulateur de voilier par des essais en mer, une premiére tentative. In: Workshop Science-Voile. École Navale, LanveocPoulmic, France (2005)
9. Clarke, D., Gedling, P., Hine, G.: The application of manoeuvring criteria in hull design using linear theory. Trans (RINA) 125, 45–68 (1983)
10. Van Oossanen, P.: Theoretical estimation of the influence of some main design factors on the performance of international twelve meter class yachts. In: Presented at The Chesapeake sailing yacht Symposium, Annapolis, Maryland, January 20 (1979)

A New Class for Robotic Sailing: The Robotic Racing Micro Magic

Alexander Schlaefer, Daniel Beckmann, Maximilian Heinig, and Ralf Bruder

Abstract. A number of boat designs have been proposed for robotic sailing, particularly inspired by competitions like the Microtransat challenge, SailBot, and the World Robotic Sailing Championship. So far, most of the boats are one-offs, often highlighting naval architecture aspects. We propose a new one-design class based on a readily available kit. Small, lightweight and with proven sailing performance the robotic racing Micro Magic presents a more standardized alternative, particularly for algorithm development and multi-boat scenarios. Our intention is to introduce an evolving class, and we propose a set of basic rules and describe the modified boat design, electronics, sensors and control approach for our prototype. Moreover, we have used four identical boats for the past year and we present results illustrating the good and comparable sailing performance, indicating that the class is suitable to study robotic sailing methods.

1 Introduction

Competitions like the Microtransat challenge [3], SailBot, or the World Robotic Sailing Championship provide environments for evaluating and comparing the overall performance of robotic sailboats. As of now, a number of small and medium sized boats have been built by different groups. However, with the exception of the mobile ocean observation platform (MOOP) designed by a team from Aberystwyth University [12], most boats have been one-offs. This is reflected by the rather coarse design rules implied by two existing classes, the Microtransat and SailBot class. One advantage is the room to try really innovative hull and rig designs. Yet, a disadvantage lies in the fact that boat performance is hard to compare and any rating formula is

Alexander Schlaefer · Daniel Beckmann · Maximilian Heinig · Ralf Bruder
Institute for Robotics and Cognitive Systems, University of Luebeck, Ratzeburger Allee 160, D-23562 Luebeck
e-mail: {schlaefer,heinig,bruder}@rob.uni-luebeck.de, daniel.beckmann@web.de

typically seen as either unfair, or too complicated, or often both. Hence, introducing a more restricted class allowing for direct comparison of boats and methods would be interesting.

Few of the existing designs would be practical as a prototype for such a new class. The MOOPs are built for long term autonomy and durability, but lack sailing performance, particularly upwind. A number of SailBot boats have beend designed at the United States Naval Academy [8, 9]. While the sailing performance is excellent, they are currently custom built and not widely available. Moreover, the length and draft, 2.0 m and 1.5 m, respectively, make the boats harder to launch. This is even more problematic with the larger Microtransat boats, e.g., the Welsh *Beagle B* [11], the Swiss *Avalon* [5], the Portuguese *FASt* [1], or the Austrian *Roboat* [7].

As another alternative, model sailboats have been used [4, 13] and are often easier to handle and comparatively low priced. We have previously described our experience with very small one-design boats for robotic sailboat racing [2]. Based on our initial results we have refined our design based on the popular Graupner Micro Magic kit. We propose the robotic racing Micro Magic (rrMM) as a new class at the World Robotic Sailing Championship. This paper presents the motivation for establishing the new class and discusses the proposed class rules. Moreover, we summarize the hard- and software design to enable other teams building an rrMM-class boat. Finally, we give initial results with respect to the sailing performance and the overall handling of the boat.

2 Class Rules

Our key motivation to study one design boats was to establish a testbed for algorithm development, i.e., the sailing performance should mostly depend on the methods used to control the boat. A further objective was a small size so that the boat can be easily carried and launched, even on smaller lakes. Moreover, most parts should be readily available and simple to assemble, as the focus is on control algorithms and not on naval design. Furthermore, the cost for parts and components should be low to encourage a wide adoption of the class. Finally, the boat should be sailable in a sufficiently wide range of conditions to allow frequent testing without depending too much on the weather conditions.

We found the Graupner Micro Magic kit to be a good compromise with respect to the aforementioned criteria. The boat is small, inexpensive, and widely available. Having built a fleet of five boats we feel confident that the Micro Magic presents a good basis as an entry level robotic sailing class. One of our modified rrMM boats is shown in Figure 1.

While a strict one design would be ideal to compare the performance of the algorithms, boats need to be adapted to different conditions to sail efficiently, e.g., by weight trim or reefing. We therefore deliberately allow some degrees of freedom in building and trimming an rrMM-class boat and propose the following rules as a basis for competitions:

Fig. 1 The figure shows a lateral and top view (left and center) of one of our boats. On the right hand side, a more detailed view of the deck layout and the electronics is shown. In contrast to the conventional design we have mounted the servos on deck and our control board to the lid. Both servos are sealed and a carbon fiber rod is used to connect them to the boom fittings. The black box inside the hull houses the GPS unit, while the black part in the cockpit is the GPS antenna.

1. The hull and deck must be from the Graupner kits (either MicroMagic or racing-MicroMagic)
2. The hull must be unmodified, while the deck layout may be changed (e.g., to add openings)
3. The keel fin and rudder must be from the Graupner kits (either MicroMagic or racing-MicroMagic)
4. The material, shape and weight of the keel bulb may be altered, as long as it is not harmful to the environment
5. The jib and main sail may not exceed the area of the Graupner kit sails
6. The type of rig is open, as long as the limitation regarding the sails is fulfilled and the boat fits into a sphere of 1 m diameter (excluding sensors / antennas mounted on the mast top)
7. The boat must be human controllable throughout the races
8. The boat must communicate its true GPS position to a central server at a frequency of at least 1 Hz
9. Boat control shall be completely automatic without human intervention

These rules try to trade-off the objectives of having directly comparable boats and allowing some room for improvements. To achieve the former, hull, keel, and rudder are rather tightly restricted, and the sixth rule intends preventing too extreme rig designs. However, controlling the sails is a key problem in robotic sailing, and under

Table 1 Dimensions of the Graupner Micro Magic. Note that the weight depends on the edition (standard vs. carbon) and on the additional components to control the boat. Our standard rrMM boats weigh approximately 1030 g with the standard keel and including the battery.

Dimension	Value
Length	530 mm
Beam	180 mm
Height	980 mm
Weight	950 g

the rules it will be possible to experiment with different rigs and methods to actuate the sails. The eighth rule has been added to promote collision avoidance, which we identified as another key problem, particularly in fleet races with similar boats.

According to the class rules the boats are not required to be completely autonomous, i.e., the algorithms controling the boat can be executed remotely. Advantages include less weight and power consumption onboard, the use of high performance standard computer hardware and high-level programming languages, and simplified debugging during test runs.

3 Hardware

Assembling the Graupner kit is rather simple, and we refer to the Graupner manual [6] and only briefly summarize our experience building the boat. Particularly, we will present a few modifcations to drive the sails, and then focus on electronics and sensors. More information is available online [10].

3.1 Boat

The mechanical dimensions of the Micro Magic are shown in Table 1. Hull, keel and rudder are mounted according to the Graupner instructions. However, the original Graupner design uses sheets to control the sails and while the jib is attached to a boom, the latter is connected to the deck by a small string. This complicates adding sensors to measure the sail angles and we therefore opted for a direct link between sail servos and booms. The jib boom is mounted on a small vertical pole close to the bow and both servos are mounted on deck, using a carbon rod to connect the servos with the boom fitting, see Figure 1. On the one hand, advantages include minimal backlash, direct sail position control via the servos and the ability to back the sails. On the other hand, the servos must be sealed against water, and the axis of rotation for the sails must be considered. While the main sail rotates about the vertical mast, the jib rotates about the inclined forestay. Hence, the horizontal rotation of the servo must be converted into a rotation about the stay as illustrated in Figure 2. Note that we have added a horizontal offset as shown in Figure 2b and use the ball bearing position to control how much the boom can rise, see Figures 2c and 2d.

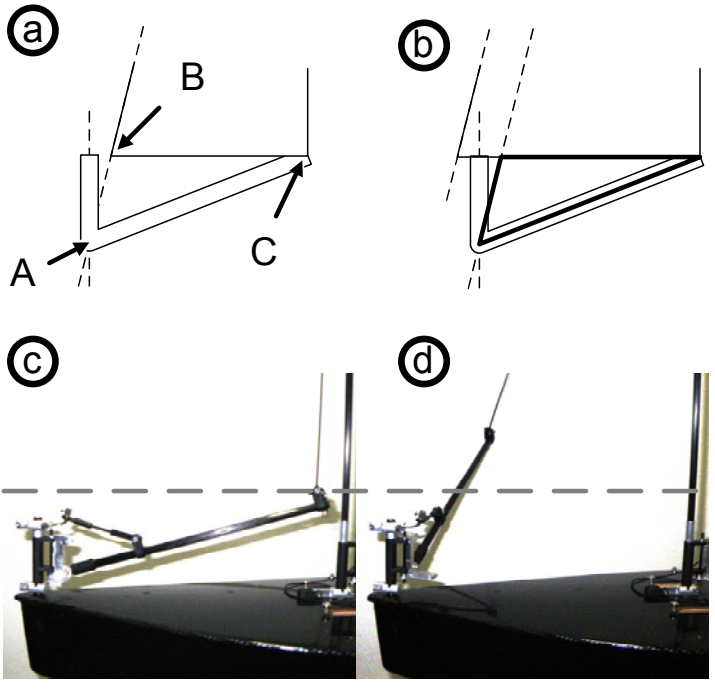


Fig. 2 The figure illustrates how the rotation from the deck mounted servos is converted into a rotation about the forestay: a) both axis are shown and the triangle ABC has to rotate about its side AB, b) the actual design adds an offset betwen forestay and axis of rotation, but the triangle is still rotating almost parallel to the forestay, c) in closed position the boom is low and keeps the leech tight, and d) when the sail is open, the boom rises to keep the leech tension. Note that by adjusting the position of point B the leech tension in the open position can be controlled.

Given the already relatively small sail area it is rather untypical to adjust the sails to the wind force, i.e., to reef. Instead, the keel can be changed to a heavier bulb, which can extend the sailable wind range to up to 5 Bft. Figure 3 shows the standard 370 g bulb from the kit, and two performace bulbs (RT-Modellbau, Germany) with 370 g and 470 g, respectively. Note that the keel position can be moved slightly forward and aft to trim the boat, but generally it is advisable to mount the heavier keels further towards the stern.

3.2 Electronics

The on-board electronics is based on an ATmega 2560V (Atmel, CA) microcontroller operating at 11.0592 MHz. The chip features built-in hardware for jitter-free, phase- and frequency-correct waveform generation, which we use to generate pulse

Fig. 3 Three different keel bulbs: a) the standard 370 g bulb, b) a 370 g performance bulb, and c) a 470 g performance bulb

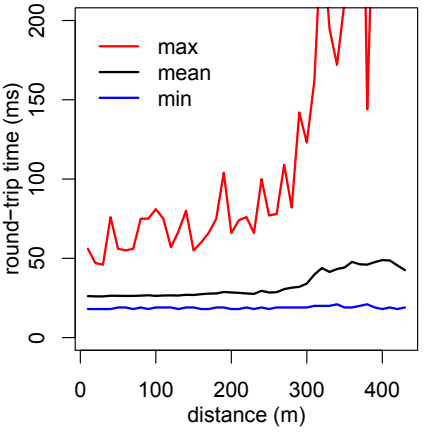


Fig. 4 The round-trip data transmission time from boat to onshore computer, indicating that the latency is below 20 ms for typical operating distances of the boat

width modulation (PWM) signals for the three RC servos moving rudder, main sail and jib. Likewise, the microcontroller is also reading and preprocessing the sensor data and controlling communication with the onshore computer.

For data transmission we use long range bluetooth modules (F2M03GXA-S01) (Free2Move, Sweden), allowing for in excess of 500 m line of sight distance [2]. Clearly, to get this range when communicating with an onshore computer, a similar module and a high performance antenna must be used, too. Moreover, as the distance between boat and onshore antenna increases, the likelihood for transmission errors and re-transmission is also rising. This can cause additional latencies, as illustrated in Figure 4. While the average round-trip time remains below 40 ms for up to 300 m distance, the communication may also be affected by other boats, e.g., when disrupting the line of sight. Considering additional latencies due to the serial link, message processing, and the timer interval, the total latency is typically below 180 ms and could be further reduced by shortening the timer interval.

3.3 Sensors

Autonomous control requires reliable sensors, which additionally need to be small, lightweight and energy efficient to be used with the rrMM. We equipped our boat with a number of relative and absolute sensors, i.e., wind speed, wind direction, 3 axis compass, 3 axis accelerometer, 3 axis gyroscope, and GPS.

For inertia measurements we used the combined magnetometer and accelerometer iM-3501 (Amosense, Korea) and the gyroscope ITG-3200 (Invensense, CA). Both sensors are interfaced using the I2C protocol. The IM-3501 provides calibrated three-axis magnetic field and three-axis acceleration measurements at a frequency of

50 Hz. Furthermore, an on-chip algorithm computes tilt-compensated compass data with azimuth and roll/pitch resolutions of one degree. The ITG-3200 is a three-axis MEMS gyroscope, providing calibrated rotational measurements with a resolution of 1/14 degrees at measurement frequencies of up to 2000 Hz.

The wind speed and wind direction sensors are custom made based on the popular AS5040 sensor (Austria MicroSystems, Austria). Finally, for absolute position and speed over ground we used the *ublox LEA4-T* GPS chipset (ublox, Switzerland) mounted on a custom designed board. The GPS unit connects to the main controller via a serial interface using the binary UBX protocol.

4 Control

As indicated before, the onboard microcontroller is mainly reading and pre-processing sensor data, setting the servo positions, and running the communication. Figure 5 illustrates that the current board operating at 50 Hz for the internal sensor processing loop, and 5 Hz for the external communication loop, has a moderate duty cycle. The limiting factor would be an increase in the frequency to send data via bluetooth.

The main control runs onshore, primarily to have sufficient computing power to allow studying more sophisticated methods, e.g., for path planning and collision avoidance. While basic processing of sensor data and control of sail and rudder position is done at 5 Hz, other tasks like filtering sensor data or path planning run asynchronously in separate threads. Another advantage of the onshore implementation is simple debugging and prototyping, e.g., the software can be changed and compiled without even restarting the boat. Moreover, essentially any high-level language can be used for development and we are currently having prototypes implemented in Java and C#.

Although we are just starting to analyze log-files from test runs, we have identified an interesting aspect of sail control. The conventional racing Micro Magic is

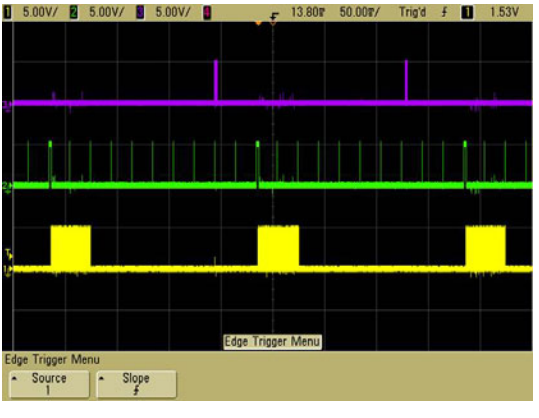


Fig. 5 The sensor data processing loop operating at 50 Hz (green), with the command queue executed at 5 Hz. Frequency and processing time for sending and receiving messages are shown inpurple and yellow, respectively. The yellow signal also indicates that the external frequency could be increased to approximately 20 Hz.

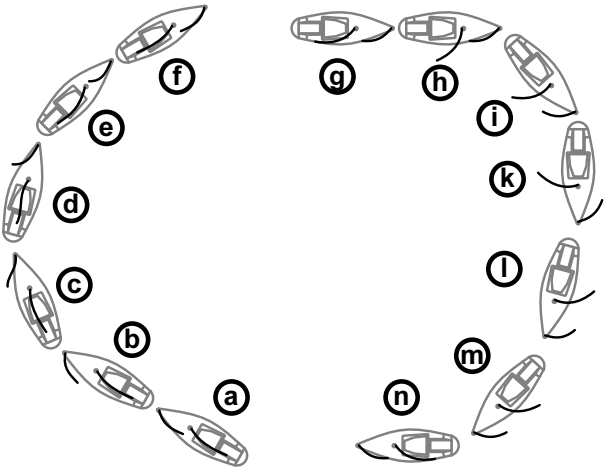


Fig. 6 The figure illustrates how jib and main sail are moved independently during maneuvers. On the left side, a typical tack from a close-hauled on starboard tack to a close-hauled on port tack is shown (a-f). Note that the jib is opened before turning the rudder (b), which is crucial to power through short waves. The jib is slack in (c) and starts backing in (d) to finally push the boat onto its new course (e,f). On the right side, a jibe from beam reach to beam reach is illustrated (g-n). The main sail is opened before initiating the jibe (h) and the sails are shifted individually to keep the boat stable (k,l).

typically operating both sails with a sheet, which apart from a small offset for the jib cannot be adjusted independently. We initially used to move both sails proportional to the apparent wind angle, e.g., during a tack both sails moved synchronously with the apparent wind. However, we quickly noticed that this made tacking in stronger winds and against short and steep waves of approximately 0.1 m almost impossible, as the boat would quickly come to a complete stop and loose rudder control. Slacking the jib before initiating the tack and maintaining an offset until the boat is on a close-hauled course on the new tack improved the tacking substantially, compare Figure 6a-f. Similarly, it is virtually impossible to have the boat fall off in stronger wind unless the main is slacked first, as shown in Figure 6g-n. This highlights that individual sail control and the ability to back the sails can help control the boat.

5 Results

Our current design is intended as a prototype for the class, and we expect future improvements to virtually all aspects. Here we summarize some basic results, e.g., regarding the robustness of the sensors and the control board, and with respect to the sailing performance.

5.1 Robustness

As size was one criteria when selecting the Micro Magic as the basis for an one-design class we can report that handling the boat is indeed very simple. We frequently carry up to four fully rigged boats in one car, and the boats are small and lightweight enough to fit into standard luggage. The boats are also very robust, on-shore and while sailing. We have sailed our boats in winds from 2 kn to 20 kn.

Overall, the electronics are also quite stable with GPS and bluetooth active even when the boat is heeling. The biggest issue is the compass sensor, which even after a calibration as recommended by the manufacturer shows systematic errors. More importantly, the sensor seems to be sensitive to fast rotations, e.g., when the boat is heeling or sailing in short waves. Figure 7 summarizes compass versus GPS plots for four different boats (yellow, green, red, and blue). Clearly, surface currents and wind do have an impact, as does the delay in the GPS heading, but the plots illustrate

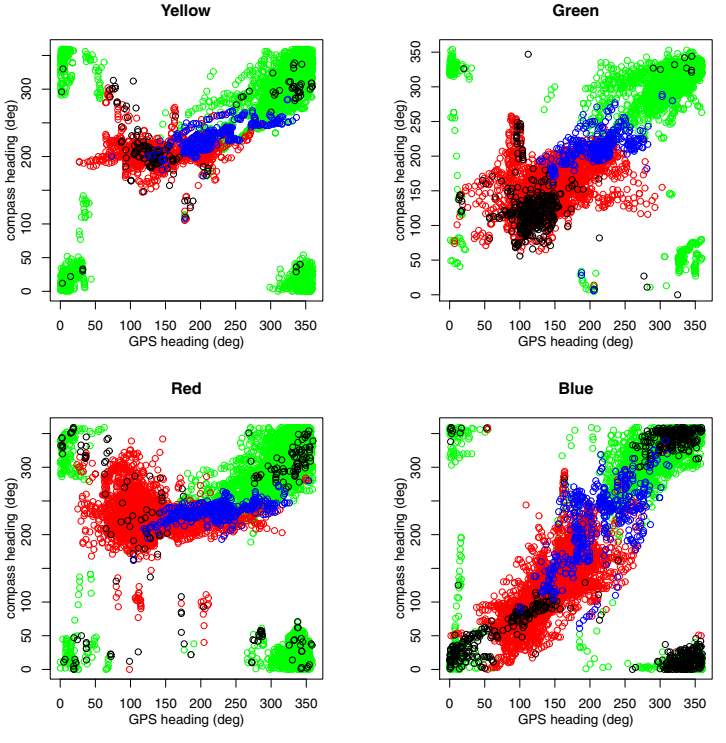


Fig. 7 A comparison of the compass heading with the GPS heading for all four boats in similar conditions. Red and green indicate values on starboard and port tack, respectively, i.e., with the apparent wind angle from 30 - 150° or 210 - 330°. Black and blue indicate an apparent wind angle from 330 - 30° and 150 - 210°, respectively.

that a careful calibration is required and also indicate that the quick motion of the boats needs to be considered when processing the compass readings.

5.2 Performance

In order to obtain data for different courses, the boats were set up to sail in a rectangle for which the longer side was aligned with the wind direction. The rudder was controlled to maintain a constant target apparent wind angle, which was changed in a round robin fashion. First, discrete wind angles with 10 degree increments were partitioned in windward and leeward courses. Second, two states were defined, direction and tack. Whenever the boat crossed the windward side of the rectangle towards the wind, the direction was set to be leeward. Likewise, when the boat crossed the leeward side to the leeward, the direction was set to be windward. Similarly, the tack was set to starboard tack when the boat moved to the right of the right side and to port tack when the boat moved to the left of the left side. Third, depending on the direction, a new target apparent wind angle was selected from the respective partition every 20 s.

To obtain a polar plot summarizing the speed with respect to the apparent wind angle, the GPS speed and apparent wind angle were sampled with 5 Hz and the minimum speed as well as mean and standard deviation of the apparent wind angle were computed for a sliding window of 50 samples. The data was stratified by the apparent wind angle and restricted to episodes where the minimum speed was larger than 0.5 m/s. The actual experiments where done using four almost identical boats and sailing in moderate conditions of approximately 3 Bft. Figure 8 summarizes the

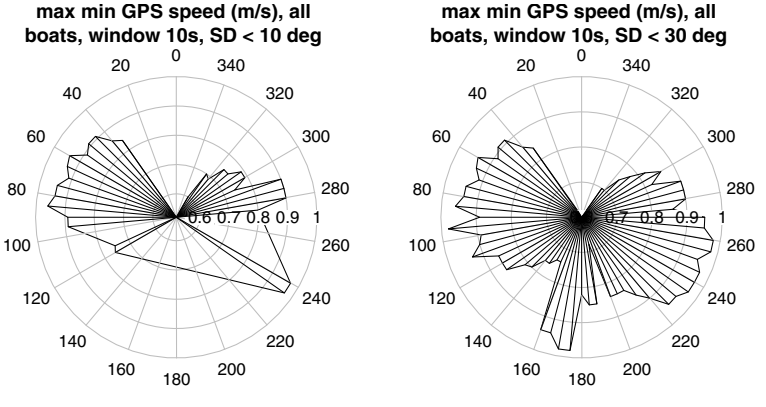


Fig. 8 An illustration of the boats' sailing performance in winds of approximately 3 Bft as polar plots. The plots show the maximum of the minimum speed in a 10 s window for each apparent wind angle. For the left plot, the standard deviation of the apparent wind angle was restricted to be smaller then 10°, i.e., episodes with a larger standard deviation where excluded. The right plot relaxes the latter requirement to 30°, indicating that it is particularly hard to maintain a constant apparent wind angle on a downwind course.

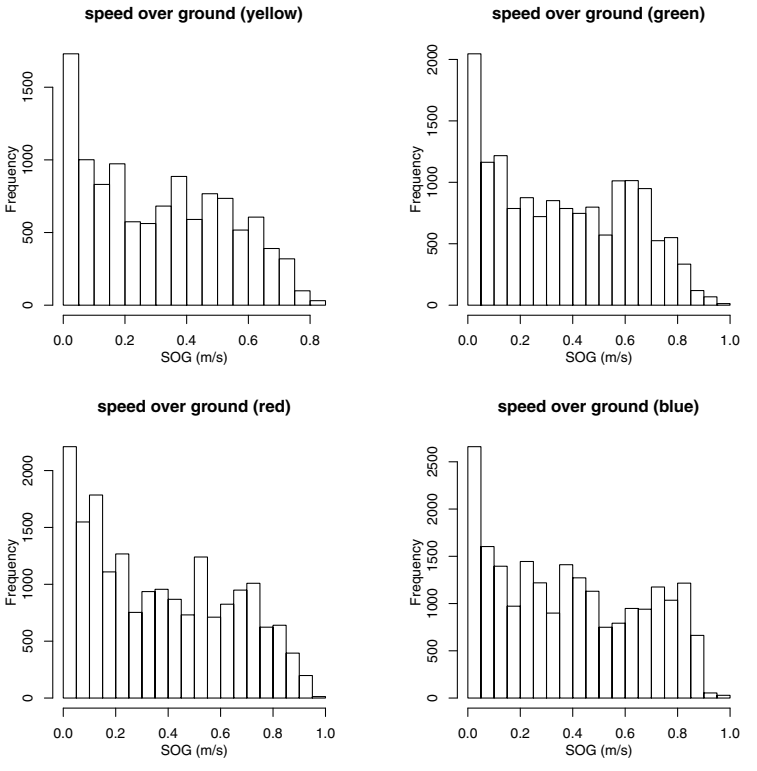


Fig. 9 Histograms showing the typical boat speed distribution during the tests. The plots indicate that the boats reach comparable speeds. Note that the yellow boat started the course later and the wind calmed down towards the end of our test.

results. Unfortunately, we cannot expect the apparent wind angle to stay constant throughout the sampling windows, e.g., due to residual errors in the boat control, but also due to changes in wind direction and waves moving the mast and the wind vane. To account for this, we restricted the standard deviation for the apparent wind in the sampling window. The left polar plot shows that upwind we can get fairly stable courses, while the right plot indicates that it is particularly hard to maintain a constant apparent wind angle on a downwind course. Furthermore, the results show that the rrMM is capable of sailing at sustained speeds of almost 1 m/s, or 2 kn.

As the boats are intended to have similar sailing performance, we also summarize the boat speed for all four boats (Figure 9). Clearly, the histograms give only a rough idea of the sailing performance, but given that the boats where certainly not perfectly trimmed, the results indicate no fundamental difference in boat speed. Note that not all boats where sailing at exactly the same time and the wind speed decreased towards the end of our tests. The boats started in the order blue, red, green, and yellow, and the green boat had to abandon the test after approximately 20 min.

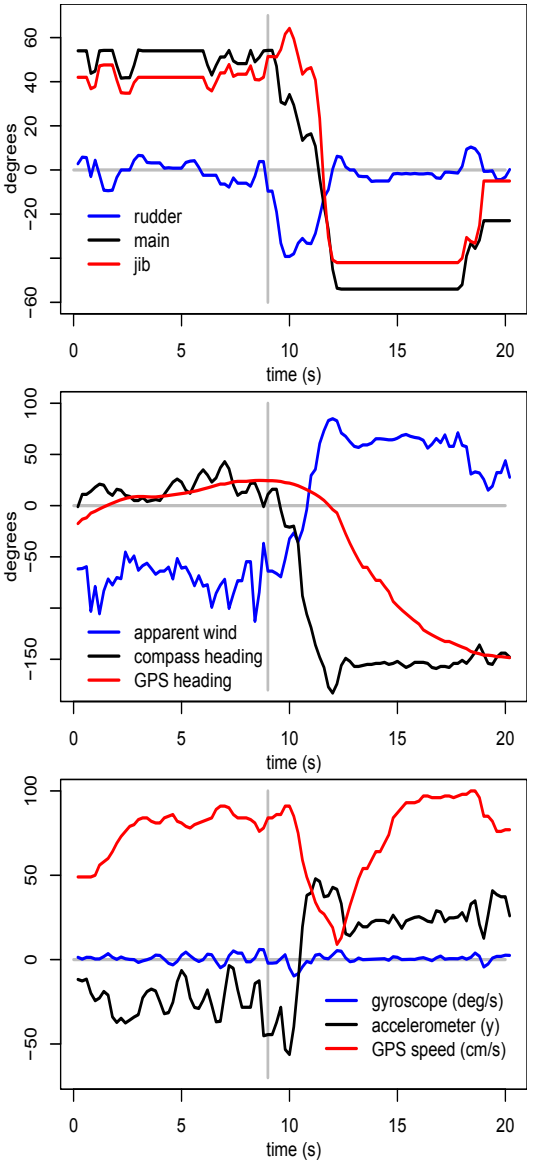


Fig. 10 Servos positions and sensor readings during a typical tack. The top plot shows the angles of rudder, main sail, and jib over time, with the vertical gray line indicating the approximate initiation of the tack. Initially, the boat is on a stable course, and the main sail angle is approximately 15° larger than the jib angle. During the tack, the rudder turns the boat, the main sail moves along with the apparent wind, and the jib angle first increases and then maintains an offset until the boat heads straight into the wind. This can be seen in the center plot, where the blue line shows the apparent wind angle. The compass readings also change immediately, while the GPS heading is subject to a substantial delay. A similar delay can be seen in the boath speed in the bottom plot, which also indicates that the boat did almost come to a complete stop. The accelerometer readings indicate that the boat also heeled substantially, which is likely exacerbated by backing the jib. However, the boat is on a stable course on the new tack after approximately 4 s.

Finally, Figure 10 illustrates the tacking performance of our prototype. The figure presents a situation during the aforementioned tests where the target apparent wind angle was set to 60°. Starting on a stable course with virtually no rudder movements and the main and jib at angles of approximately 55° and 40°, respectively, the tack was initiated at about 9 s. The top plot clearly shows how the rudder turns the boat while the main sail moves along with the apparent wind. Concurrently, the jib angle

first increases and then maintains an offset until the boat heads straight into the wind. This is shown in the center plot, which also demonstrates that the compass readings change immediately and the GPS heading is subject to a substantial delay. The delay also affects the speed over ground given in the bottom plot. Apparently, the boat came to a complete stop during the tack, which took approximately 4 s from stable course to stable course.

6 Conclusion

We have proposed a novel one-design class for robotic sailing. Key features of the rrMM are the small and lightweight design with a relatively inexpensive kit readily available. Based on the kit and the description of our prototype we expect that it would be relatively easy to build an rrMM class boat. Moreover, despite its size the boat can be sailed in a fairly wide range of conditions of up to 5 Bft, allowing to test algorithms on virtually every lake or in otherwise sheltered waters.

Our results regarding the sailing performance underline that the rrMM class presents a reasonable platform for development of robotic sailing methods. The boat speed of upto 2 kn allows to quickly cover a race course or to study some planning scenario, particularly as the boat can sail relatively close to the wind. Essentially, if algorithms work with an agile and dynamic boat like the rrMM, it can be expected that they could be ported to larger boats.

Moreover, programming and testing is simplified by the option to use onshore computers for boat control. Although this implies the boats are no longer strictly autarkic, they are still autonomous in the sense that they are fully computer controlled. Hence, the new class presents a viable alternative for groups interested in testing new approaches, particularly for multi-boat scenarios.

However, we do not see the rrMM class as a replacement for the SailBot and Microtransat classes. Instead, it complements these classes, both, in presenting an affordable and small entry level platform and by focusing on algorithmic aspects. The latter also follows from the class rules, which limit the differences between boats. Yet, the proposed rules allow for some flexibility, e.g., in principle permitting swing or wing rigs. While we expect the rules and the design to evolve over time, we hope that other groups will adopt the rrMM for their projects.

References

1. Alves, J.C., Ramos, T.M., Cruz, N.A.: A reconfigurable computing system for an autonomous sailboat. Journal of the Osterreichische Gesellshaft für Artificial Intelligence (Austrian Society for Artificial Inteligence) 27(2), 18–24 (2008)
2. Ammann, N., Biemann, R., Hartmann, F., Hauft, C., Heinecke, I., Jauer, P., Krüger, J., Meyer, T., Bruder, R., Schlaefer, A.: Towards autonomous one-design sailboat racing: navigation, communication and collision avoidance. In: 3rd International Robotic Sailing Conference, Kingston, Ontario, Canada. Ammann, N., Biemann, R., Hartmann, F., Hauft, C., Heinecke, I., Jauer, P., Krüger, J., Meyer, T., Bruder, R., Schlaefer, A, pp. 44–48 (2010)

3. Briere, Y., Bastianelli, F., Gagneul, M.: Challenge microtransat. In: CETSIS, Nancy, France (2005)
4. Bruder, R., Stender, B., Schlaefer, A.: Model sailboats as a testbed for artificial intelligence methods. In: 2nd International Robotic Sailing Conference, Matosinhos, Portugal, pp. 37–42 (2009)
5. Giger, L., Wismer, S., Boehl, S., Büsser, G.A., Erckens, H., Weber, J., Moser, P., Schwizer, P., Pradalier, C., Siegwart, R.Y.: Design and construction of the autonomous sailing vessel avalon. In: 2nd International Robotic Sailing Conference, pp. 17–22 (2009)
6. GRAUPNER GmbH & Co. KG, K.G.: http://www.graupner.de/fileadmin/downloadcenter/2014.C_racing_MM_Carbon_Edition_de_en_fr.pdf (cited May 31, 2011)
7. Klinck, H., Stelzer, R., Jafarmadar, K., Mellinger, D.: Aas endurance: An autonomous acoustic sailboat for marine mammal research. In: 2nd International Robotic Sailing Conference, Matosinhos, Portugal, pp. 43–48 (2009)
8. Miller, P., Beal, B., Capron, C., Gawboy, R., Mallory, P., Ness, C., Petrosik, R., Pryne, C., Murphy, T., Spears, H.: Increasing performance and added capabilities of usna sail-powered autonomous surface vessels (asv). In: 3rd International Robotic Sailing Conference, Kingston, Ontario, Canada, pp. 57–62 (2010)
9. Miller, P., Brooks, O., Hamlet, M.: Development of the usna sailbots (asv). In: 2nd International Robotic Sailing Conference, Matosinhos, Portugal, pp. 9–16 (2009)
10. rrMM webpage, http://www.r2m2.org (cited May 31, 2011)
11. Sauze, C., Neal, M.: Design considerations for sailing robots performing long term autonomous oceanography. In: International Robotic Sailing Conference, Breitenbaum, Austria, pp. 21–29 (2008)
12. Sauze, C., Neal, M.: Moop: A miniature sailing robot platform. In: Proceedings of the 4th International Robotic Sailing Conference (2011)
13. Stelzer, R., Pröll, T., John, R.: Fuzzy logic control system for autonomous sailboats. In: IEEE International Conference on Fuzzy Systems (2007)

Part III
System Development

A Systems Engineering Approach to the Development of an Autonomous Sailing Vessel

Bradley E. Bishop, Joseph Bradshaw, Cody Keef, and Nicholas Taschner

Abstract. Over the past four years, undergraduate students in the Systems Engineering degree program at the United States Naval Academy (USNA) have pursued autonomous sailboat systems development as part of their required capstone project in conjunction with students from the Naval Architecture program that design and fabricate the vessels. In this paper, we discuss the pedagogy of robotic sailing as a capstone for Systems Engineering students and discuss our approach to interdisciplinary tasks such as this. We also outline the design philosophy associated with the systems side of the equation, focusing on the on-board instrumentation and control hardware.

1 The Pedagogy of Robotic Sailing Competitions

The United States Naval Academy (USNA) is a baccalaureate institution that provides to every student both a degree and a commission in the United States military, primarily the Navy or Marine Corps. Students in the Weapons and Systems Engineering department are required to complete a capstone project as part of their ABET-accredited degree in Systems Engineering. The students take a semester course on project management in the Fall semester of their 1/C (senior) year. In this course, students propose a project and are assigned a faculty advisor. In the following (Spring) semester, students execute their plan and produce a fully-integrated capstone project with a formal performance and design review.

Systems Engineering at USNA focuses on mechatronics, robotics and feedback control. As such, our projects tend to be very hardware-intensive and must include some form of autonomy or automation. Due to the marine-centric nature of our service, many students opt to work on projects that involve autonomous marine vehicles, both surface and underwater. Many students choose autonomous sailboats for their capstone projects due to their familiarity with sailing or from a desire to apply advanced control and automation in a competitive framework.

Bradley E. Bishop · Joseph Bradshaw · Cody Keef · Nicholas Taschner
United States Naval Academy, Annapolis, MD, USA
e-mail: {bishop,bradshaw,m116552}@usna.edu, cody.keef36@gmail.com

1.1 Robotic Sailing as a Capstone Project

While autonomous sailing vessels have been developed at USNA prior to the effort discussed in this paper, the origin of this work was initially the SailBot competition [2, 3], and later the World Robotic Sailing Championship (WRSC) [4]. It has been the point of view the USNA Systems Engineering department for many years that student participation in a competition can be both a blessing and a curse. While the students are highly motivated to win, it is easy for them to lose sight of good design principles and the requirements of a capstone project in the face of the immediate demands of the competition. It is our belief that winning the competition should never be the primary goal, but rather should be a consequence of a proper design experience that begins with an appropriate needs analysis.

While it seems logical that the winning entry in a competition will of needs be the best designed of the field, it is not necessarily the case. Stop-gap measures, kludges, and inelegant coding, wiring and design are the hallmarks of projects whose only metrics are victory at any cost. While one of these vessels may win a competition that spans a few days, they will often have deep design flaws that may not be obvious from their performance. These flaws in design are the hallmarks of a failed capstone project: one in which the students did not learn the appropriate lessons of design and built an overly-inflated view of kit bashing and caffeine-fueled desperation that will not stand the test of a real-world engineering problem.

It is therefore essential that students engaged in a capstone design whose output is to be used in a competition be guided appropriately through the proper engineering design, and be evaluated on the same. This is a problem of perception on the part of the students and expectation on the part of the advisor(s) and the host department, both of which need to be managed carefully from the outset of the project.

1.2 Project Execution Methodology: SailBot

SailBot is a student-driven competition involving automated sailboats that compete in racing, navigation and station-keeping contests. SailBot class vessels are 2 m or less in length, while Open Class vessels can be up to 4 m in length. Full details of vessel requirements can be found in [2].

It is important to note that the SailBot project at USNA is an interdisciplinary effort between the Systems Engineering department and students pursuing a Naval Architecture degree. The Naval Architecture students design and build the vessel in a two-course sequence (Fall-Spring), and the Systems Engineers provide instrumentation, actuation and automation as a follow-on effort the next academic year. This process has yielded quality results for many years now, and there are important lessons that have been learned. In this section, we will discuss the formal pedagogy of SailBot as a capstone design project for the Systems Engineering students, starting with a completed but strictly R/C vessel generated by the Naval Architecture students.

Students in Systems Engineering at USNA are expected to formally define a full set of objectives, metrics, functions and demonstration plans for their design projects [1]. Through the remainder of this section, we will discuss an appropriate scope for the SailBot project and lay out the objectives and performance metrics. For purposes of this discussion, we will assume that the problem and needs statements for the design are completely defined by the fixed vessel (provided by the Naval Architecture students) and the SailBot competition rules [2, 3]. Further, there are constraints on weight and on size/volume of each component that come from the fixed vessel design and must be discussed with the Naval Architecture students. It is impossible, for example, to place a standard laptop PC into the SailBot designs that have thus far been generated at USNA due to the access hatch size and hull form, so selecting such a system is *a priori* prohibited.

1.2.1 Objectives and Functions

The first step in a proper system design, after a clear needs analysis, is to lay out the objectives of the design. An *objective* for a design is one of a set of characteristics that differentiate a *good* design from a bad one [1]. Objectives describe what the system *is*, not what it *does*, and a fully defined objectives tree is a crucial step toward developing a good project that meets the needs of the problem.

Students typically struggle to generate the appropriate depth and breadth of objectives for a complex project. Wanting to focus on what the vessel does (functions) and how it will accomplish those functions (means), students skip over the vital step of defining design objectives. The task presented to the students is to define the objectives and then push toward the functions that support those objectives. The following is a discussion of the primary objectives that are crucial to a good SailBot system design (in bold), with the high-level functions that are needed to support them. There are some additional design objectives and further refinements that are not listed here due to space restrictions.

The system must *be*:

1. **Competitive:** this objective focuses on the tasks that are required for victory in the SailBot competition. The vessel and its controller must be:
    a. **Fast:** the system must achieve its navigation and sailing tasks quickly in order to compete.
    b. **Accurate:** the system must minimize error in navigation and in executing selected sailing commands.
    c. **Energy-efficient:** the system must not waste battery power, as some of the tasks are long-duration.

2. **Reliable:** the systems must be designed to reduce maintenance and update costs (in time and money):
    a. **Robust:** the system must be capable of carrying out its mission with minimal sustained damage over many cycles. Must be concerned with

temporary power loss, water intrusion, damage from slamming (waves), and possible collisions.

b. **User-Friendly:** the system must admit quick setup, configuration, power recharge, component replacement and in-situ reconfiguration.

Note that the full set of functions is not defined here, but there are several functions described above that would not be clear from the basic SailBot task. Students must fully enumerate the set of functions while avoiding solution bias once the full set of objectives is defined.

The set of functions (things that the system must *do*) is given by the implicit functions above as well as the following:

1. **Allow configuration of task:** the software or hardware must allow a user to select from a suite of tasks (station keeping, match racing, etc.) *and* configure the parameters of the task (maximum allowable deviation, GPS coordinates, etc.).

2. **Compute sailing commands:** the system must be able to determine appropriate sailing commands to control the available surfaces (typically, for our vessels, the rudder and the main sail).

3. **Carry out sailing commands:** the system must execute sailing commands, meaning that it must actuate the surfaces at the appropriate time and to the appropriate position.

4. **Measure appropriate vessel states:** the system must measure data as required by the sailing command generation subsystem. The exact nature of the data required should not be specified at this point.

5. **Avoid obstacles:** the system must recognize and avoid navigation obstacles, including other vessels and structures that might damage the vessel.

6. **Provide remote override:** the system must allow a human to remotely take control of all sailing functions in order to maintain safety.

7. **Transmit data:** the system must provide telemetry to the shore, indicating all data gathered and computed sailing commands.

8. **Provide power:** the system must provide power sufficient for all subsystems for the specified operational period.

1.2.2 Demonstration Plan and Metrics

The set of design objectives and functions above are the hallmarks of a good design, but they are vague and open to interpretation. Students must be provided with a method by which they can test and validate their design in a rigorous and meaningful way. It is here than many competition-based projects run into trouble. Specifically, it is inappropriate to design to a metric that cannot be evaluated

during the design process. If victory at the competition is the only performance goal, it will be impossible for the students to carry out the iterative design of the individual components and subsystems that is required by a quality design process. As such, students pursuing SailBot at USNA must define a set of metrics by which their system can be assessed *before* the competition. This set of metrics must be driven by the tasks of the competition, and realistic performance levels must be enumerated (we use a 0 – 4 system, with 4 being the best result and 0 being unacceptable).

The following describe a full set of metrics for the primary objectives of the SailBot system, with the point values for each level of performance:

1. **Fast:** given a change in environment state or task configuration, the system with compute and reach a new course in:

   4 points: less than 5 seconds
   3 points: between 5 and 7 seconds
   2 points: between 7 and 9 seconds
   1 point : between 9 and 10 seconds
   0 points: greater than 10 seconds

2. **Accurate:** passes within a specified distance, on a specified heading, of a target GPS point (corresponds to scoring buoys in races). The best score is achieved when passing inside the start marks, which are 3m apart. There are further marks an additional 3m beyond those start points:

   4 points: passes within 1m of target
   3 points: passes within 1-1.5m of target
   2 points: passes within 1.5-3m of target
   1 point : passes within 3-4.5m of target
   0 points: greater than 4.5m from target

3. **Energy-efficient:** the system must provide energy for endurance trials. The scoring rubric is battery life (in hours) divided by 2, with a max score of 4 (8 hours of battery is desirable).

4. **Robust:** this metric scores based on 1-point binary decisions for a total score of 0 - 4:
   +1: system automatically reboots and restarts configured task when power is interrupted
   +1: system components are watertight to 6”
   +1: system components are secured against shock (mean time between failures in normal operation of greater than one day)
   +1: system can function in any navigable weather condition

5. **User-Friendly:** again, this metric scores on a sequence of one-point binary decisions for a total of 0 – 4 points:

   +1: allows wireless programming and reconfiguration
   +1: allows for battery replacement underway

+1: allows for one-point disconnect of any device or subsystem
+1: allows for one-point access to any subsystem or component

It is well and good to define a set of metrics, but it is necessary that the system be designed to meet those metrics, and that there be a plan for testing the performance. We utilize the pairwise comparison chart (PCC), morphological chart and decision matrices to select components to achieve the functions and hopefully meet the metrics [1]. Briefly, the PCC provides a weighting for each objective, relative to each other objective. The morphological chart enumerates possible methods by which each function can be achieved (e.g., sensing could be a GPS and an IMU or it could be a weather station; there might be sensors for sail pressure or for relative wind speed; etc.). A decision matrix is generated for each subsystem by taking each potential implementation for that subsystem and scoring it based on the appropriate metrics. An overall score for each possible solution is generated based on the scores and on the appropriate objective weightings. The potential solution with the highest composite score for each subsystem is the preliminary design. Details of this process are beyond the scope of this work, but can be found in [1].

The advisor plays a crucial role in the development of the preliminary design. Students will tend to gravitate toward well-understood or proven components to fill out the morphological chart, and will often resort to nonsensical entries to fill out a row. For example, when selecting components for the power system, a typical morphological chart will show: batteries (unspecified) and solar, with the occasional internal combustion or nuclear power option. Students must be guided to look into battery technologies and study their characteristics to select an appropriate suite of choices using the metrics and constraints.

Once a realistic set of options have been enumerated, the students must rely on the available data and their engineering insight to select the preliminary design using a decision matrix. Here, the advisor must assist with realistic scores for metrics that may be difficult to predict. Whenever possible, direct experience with the systems under consideration is appropriate. In 2011, the SailBot systems team developed, constructed and tested two completely different and independent sensing systems in order to evaluate the efficacy of each.

Having selected a preliminary design for each subsystem, a compatibility check is run. If two subsystems are incompatible, a decision must be made as to which to disallow. Assuming that such a decision can be made, a replacement for one or more of the incompatible systems is selected, starting with compatibility with the remaining system as a prerequisite. The process is carried out iteratively until a complete and compatible preliminary solution is obtained.

Finally, students must generate a demonstration plan by which they will test each component of the system and measure the actual performance. Again, guidance by the advisor is crucial during this step, as students will tend toward the "build the system and test the whole thing" approach. Students who work on SailBot are required to carry out the following tests on every subsystem:

1. **Component bench test.** Using a stable (wall) power supply and the least-complex interface available (manufacturer-provided testing software, simple RS232, etc.), demonstrate that the performance of the component meets its specs. This will often involve generation of test inputs and parameterized test runs, and will often result in extensive troubleshooting and (sometimes) a return to the decision matrix for an alternative solution.

   E.g.: when testing a wind sensor, students provided power from a regulated wall source and mounted the sensor on a gimbal used for testing IMUs in our autonomous vehicles class. Students provided a reasonable airflow using a shop fan and looked at the system output on a dummy terminal using HyperTerminal as the attitude of the sensor was adjusted on the gimbal. The results of the test showed that the system could reliably measure airflow with a relative angle of up to 30 degrees. The test did NOT, however, show that the measurements were accurate. A good bench test would have a means of verifying the measured data.

2. **Interoperability test.** Here, each functional component or subsystem that interacts with any additional component(s) is tested with each. This is done first with each communicating component individually and then by adding components/subsystems individually (as is reasonable and achievable… extremely complex systems have a combinatorial explosion in this test and will require additional care in selecting subsystems to test). When and if failures occur, individual components can be isolated for further analysis and troubleshooting. This set of tests is only complete when the entire, fully-connected system is functional with stable wall power. It is crucial to point out that the system does not need to be fully programmed to test the interoperability, but that any and all communication modalities and control signals must be exercised to show their operation. It is typical to start with the processor in this test, adding peripheral components one at a time.

3. **Powered tests.** Using the designed power supply, each component is again tested to show full performance is met. This step is optional and may only be required if the multi-system powered test fails (see below).

4. **Multi-system powered test.** Additional loading to the power supply is provided by bringing up each subsystem that has passed the powered test, one at a time, and verifying that each maintains its performance and interoperability.

5. **In-situ test.** The components are installed and again tested.

6. **Full system trials.** Here, each capability of the system is tested as well as possible with variables removed. Navigation tests are conducted on land, moving the vessel on a cart. This is followed by on-water tests of basic GPS-based navigation, then maneuvering (tacking, etc.), and so forth, building up sophistication.

When students follow the design process, they will inevitably run across a subsystem that does not meet specs. This is where the iterative nature of the design becomes apparent, and where all of the preliminary work pays off. Students have fall-back options in place for any subsystem that fails, and can refine decisions and metrics based on experience in the testing phase. Without this careful and systematic approach, students are often left accepting less than desirable performance because they simply cannot determine where things have gone awry and, in most cases, are not prepared to generate an alternative solution even if they did understand the primary failure mechanism.

1.3 Pedagogy of Robotic Sailing

The means by which students carry out their project are an end of sorts of their own, providing students with experience and insight into development of a complex system. It is unlikely that any student will be building robot sailboats for a living at any point after graduation, so the main outcome of the effort must of needs be a deeper understanding of the engineering process in general. It is true, however, that robotic sailing is a highly specialized task that involves all of the disparate aspects of systems engineering, from power through expert control. As students progress through the tasks associated with this project, there are several key points at which domain-specific lessons can be driven home, tying the design experience more closely to the students' core discipline and coursework.

Functional Blocks: Because the task of generating a complete sailing robotic system in just one semester is quite daunting, the planning phase becomes of vital importance. Students cannot afford the luxury of working together on every aspect of the task, and so must divide the work. Because of the tightly integrated nature of the systems, each sub-team must understand and work to a specific set of interface and power requirements. As such, every subsystem is managed as a 'functional block' in the overall system. Each sub-team leader is responsible for maintaining an accurate list of input requirements, output format, power requirements, and total volume and weight.

Budget: Because the vessels we use are optimized for sail racing, the budget aspect of the systems design goes well beyond the understood monetary level to include weight, power and volume. Because total battery capacity, weight, and volume are often related, tradeoffs must be made to select appropriate batteries. These decisions can impact available sensing and actuation technologies. Design to a four-part budget is a great exercise in optimization and planning.

Communication: Because this is a multi-disciplinary team, comprising systems engineers as well as naval architects, communication is especially challenging. The two disparate engineering disciplines do not necessarily speak the same language nor mutually understand the requirements of their distinct tasks. As such, a continual dialog is necessary. To accomplish this, there are advisors from both disciplines on the project, and representatives from the Systems team are required

to attend meetings of the Naval Architects. Further, during a Fall semester course offered to the Naval Architects, the Systems advisor gives an overview of the basics of the automation and control of the previous year's vessel and explains some of the common issues related to the systems and naval architecture interaction.

Planning: To execute a project of this nature in only one semester takes a commitment to the task and a solid plan. Students can be taught the importance of planning as this project progresses, especially if the deliverables include a full set of test results as discussed previously. When a week-by-week plan of action and milestones (POA&M) is completed as part of the preparatory course, students do not appreciate how much effort each of their tasks will take. We require our students to update the POA&M every week, and provide discussion on their progress. This allows the advisor to show the students how to generate optimality in the task order and how to better estimate actual required time. This is a skill that we find cannot be taught in a lecture, but which students rapidly appreciate when the realities of a complex task loom large and their grade is in the balance. To assist us at USNA, there are actually formal marking periods at the 6 and 12 week points in the semester, offering good targets for intermediate deliverables and maintaining a steady pressure on the students across the time of the project.

2 System Architecture of USNA SailBots

As mentioned, all of the vessels used by the USNA team for SailBot and WRSC are designed and constructed by students in the Naval Architecture department. Systems Engineering students primarily focus on selection and design of the power, sensing, control and communication subsystems and to some lesser extent the actuation. Over the last four years, the teams have developed a systems architecture that has proven robust and easily customizable, suited to wide variety of tasks. In this section, we will outline the basic components of the system and discuss the unique aspects of the system design.

2.1 Basic System Architecture

As discussed previously, the systems used for robotic sailing at USNA are developed with a 'functional block' approach. The primary subsystems and associated requirements are as follows, where each subsystem is (again) managed as a functional block that is under the purview of one subsystem manager who must communicate with the rest of the team to guarantee proper performance.

Processing: This subsystem is the heart of the robot. Here are carried out all of the computations required for planning and executing the assigned tasks. SailBot requires that all processing be on-board, and the form factor of the vessels designed precludes the use of a standard laptop or netbook. Power issues and heat dissipation must also be considered in the selection process.

Actuation: Because the system is designed to allow a human to take over from shore or a chase boat at any time, the actuators are selected to be R/C friendly. As of now, only the main sail and the rudder are actuated, although actuators have been selected for independent control of the jib as well. The key concerns when choosing the actuators are power consumption, accuracy and speed. It is desirable to have actuators that have no required holding torque (using a mechanical brake or a non-backdrivable system such as a worm gear).

Communication: The system must communicate its state to a base station for monitoring and data collection, and must allow remote reconfiguration.

Control: The system must be able to take outputs from the processor and use them to manage the actuation. Further, there must be a remote control capability that is the system default (that is, when the processor fails the system automatically defaults to remote control).

Sensing: GPS, wind data, obstacle locations, etc. must be provided to the processor as needed for the appropriate algorithms.

Power: An integrated power system must be generated that will provide appropriate power to all components regardless of load. This often includes multiple supplies as well as regulation to achieve optimal balance of weight, volume and capacity.

Containment/Mounting: All system components must be securely mounted and protected from water incursions, which are common in small amounts. The components must be easily accessible and must fit through the access panels on the vessel.

Most of the components used in USNA SailBot are commercial off the shelf (COTS) products, from the R/C sail winch to the AirMar weather station used for wind measurement and GPS data. The full list of components for SailBot changes from year to year and is beyond the scope of this paper. There is one component, however, that is unique to USNA SailBot. In the following section, we will discuss that component and its implications for autonomous system control.

2.2 The NavBoard3

As mentioned, the form factor of most SailBot systems precludes the use of a laptop or netbook as the primary processor. While we have used a small form computer (Pico system) in the past, the Technical Support Department (TSD) within the Systems Engineering department at USNA has developed a custom computation and sensing board that has a small form factor, low overhead, moderate cost and high capability. The power and flexibility of this system has been proven in its use across a wide array of projects, but none so thoroughly as the autonomous robotic sailboat. In this section, we will discuss the key features of this crucial component and provide the overall design concept behind its development so that the

interested reader can adopt and adapt the design principles to suit their needs and the wider needs of the autonomous vehicle community.

The Rabbit Navigation Board version 3.0 (NavBoard3) is a programmable single board computer that functions as an embedded controller for a variety of unmanned systems requiring navigation-based sensor feedback and output control signals. The system was designed to facilitate the ever-increasing number of unmanned system related projects hosted by the Systems Engineering department, from underwater autonomous vehicles, surface vessels and ground vehicles to a variety of aerial systems including helicopters, planes, and experimental craft.

The NavBoard3 uses a Rabbit 3000 processor core module programmed in a C programming design environment called Dynamic C. The core module can be programmed with a special cable using a PC RS232 serial communication port or over a wireless interface if a special boot-loader is used (as is done with SailBot to provide in-situ reprogramming).

The Rabbit 3000 processor, although relatively slow when compared to many modern processors, offers the advantages of a highly dependable In-Circuit Debugger for development, many built-in software libraries and functions, preemptive and cooperative software multitasking capability, large on-board memory capacity of 512K bytes of Flash program memory and 512K bytes SRAM, a Real Time Clock, and a multitude of additional input and output capability. A special NavBoard3 software library gives users access to the onboard hardware via standard C function calls, allowing the user to focus on the higher level overall system implementation and control algorithms as opposed to the low level data acquisition and digital communication interfaces.

This control board encompasses a variety of sensors including a 3-axis accelerometer and 3 angular rate gyros, a 3-axis magnetic field sensing module for deriving heading, an on-board Xbee wireless communication transceiver, four servo control ports with programmable pulse width modulation, and a GPS receiver. These sensors provide the ability to collect navigation information and sense external body rates and forces for robotic platform state estimation. There are many undedicated I/O ports to which the user can interface external sensors, both analog and digital. The NavBoard3 also has a variety of external communication bus interfaces such as I2C, SPI and serial UARTs, making it easy to interface with almost any COTS product. The overall layout of the custom board is shown in Fig. 1.

The NavBoard3 is modular and easy to maintain, with most components being easily removable for replacement. The components are all themselves COTS, and the PCB is manufactured in bulk. This device uses the processor that is the core programming and control learning platform for the department, reducing the learning curve for students adopting it as their processor. The integrated IMU capabilities reduce overall system complexity for autonomous vehicles, and allow for higher-level work on the part of the students. At the same time, any and all capabilities embedded in the board can be supplanted by external devices as needed

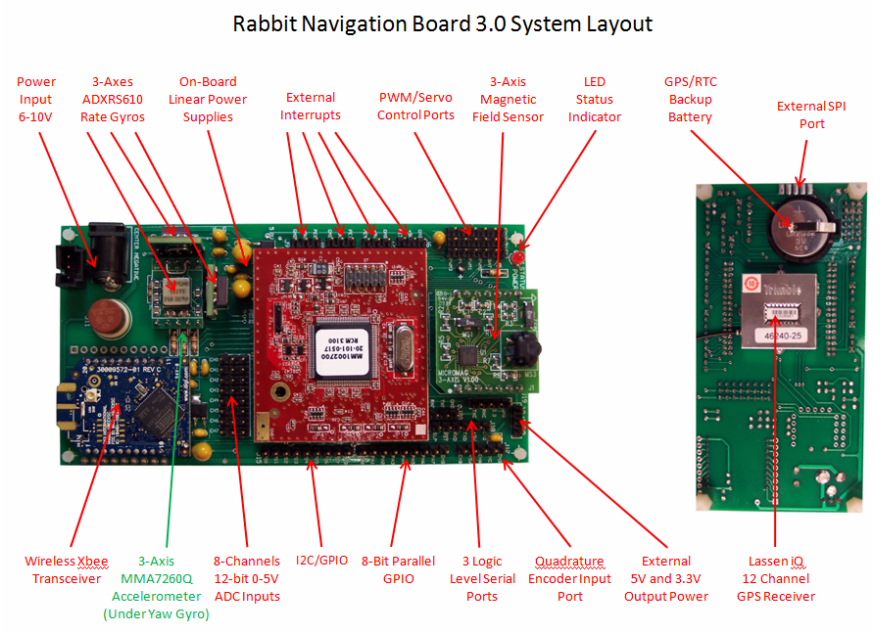


Fig. 1 The NavBoard3 layout. The board main face is shown on the left, with the view of the underside on the right. Green items indicate a component that cannot be seen in the given view (occluded by other components).

3 Conclusions

Robotic sailing is a complex, interdisciplinary task that requires significant effort in real design. While competition-driven, it is possible to achieve high-quality pedagogical results using a systematic design methodology and appropriately scoped deliverables and metrics. In this paper, we have discussed an appropriate set of design tools and tactics for system design. We have given details of our interdisciplinary interaction model, which has been developed over four years of running this project. The best recommendation that we can provide is to generate each vessel as a two-stage process, with one year of effort on naval architecture followed by a year of system design. While this is challenging, the rewards are real and substantial, and actually improve the quality of the design process.

References

1. Dym, C.L., Little, P., Orwin, E.J., Spjut, R.E.: Engineering Design: A Project-Based Introduction, 3rd edn. John Wiley & Sons, Inc., New York (2009)
2. SailBot Class Rules. SailBot: 5th International Robotic Sailing Competition (December 23, 2010), `http://www.usna.edu/Users/naome/phmiller/SailBot/SailBot.htm` (cited: May 31, 2011)

3. SailBot: Sailing Instructions. SailBot: 5th International Robotic SailingCompetition (December 23, 2010), http://www.usna.edu/Users/naome/phmiller/SailBot/SailBot.htm (cited: May 31, 2011)
4. World Robotic Sailing Championship. [Online] INNOC – Österreichische Gesellschaft für innovative Computerwissenschaften (2010), http://www.roboticsailing.org/ (cited: May 31, 2011)

Using ARM7 and µC/OS-II to Control an Autonomous Sailboat

Michael Koch and Wilhelm Petersen

Abstract. This paper presents some aspects of an autonomous sailboat project, we started at our university. An interdisciplinary group of students of mechanical engineering, electrical engineering and computer science are building the boat and are developing the control hard- and software. Our first attempt to control the boat combines an ARM7 microcontroller and the real time operating system µC/OS-II. Beside our first boat design, we will talk about the used control hard- and software, especially the mentioned controller and operating system, and the interaction between the components. We show the system architecture of the control system, the power consumption of the components and the different possibilities to process sensor data with the controller and the operating system.

1 Introduction

Working at an university located at the seaside and sailers since many years, we are searching for a student project that combines practical experience in mechanical and electrical engineering and computer science with water and sailing. In September 2010 we discussed the idea to build an autonomous sailboat with some students and now – in the actual semester – we have a small group of students, who are building the boat.

The interdisciplinary team consists of four students of mechanical engineering and four students of electrical engineering and computer science. The mechanical

Michael Koch
FH Stralsund, Electrical Engineering and Computer Science,
Zur Schwedenschanze 15, 18435 Stralsund, Germany
e-mail: michael.koch@fh-stralsund.de

Wilhelm Petersen
FH Stralsund, Mechanical Engineering, Zur Schwedenschanze 15,
18435 Stralsund, Germany
e-mail: wilhelm.petersen@fh-stralsund.de

engineering group is building the boat and the mechanical parts, while the sensors, the controller and the software are built by the computer science group. The students are supported by the workshops of both departments. They use this project as practical work within their studies. The team is completed by a student of business administration and engineering. He is organizing the needed materials and components.

The students began their work in February 2011. In May 2011, the mold of the hull of the boat is built. We hope, we can finish building the boat in July. In parallel, we are building the controller hardware and the sensors. The aim is to have a first version of the whole software in July.

2 The Boat

Our first sailbot design – called FHsailbot – is based on an AMYA[1] one meter class specification. The size of the hull is limited by the rig, especially the size of the available masts. We use a profiled aluminum mast with a groove for the sail. The maximum size of the mast (2 m) leads to a boat length of 1.52 m. The second limiting factor for our first design are the transport possibilities of our cars and trailers. To test the control system during the building the boat, we equipped an old model boat – called Saudade – with the controller, sensors and actuators. So we can test the components and the software independent of the new boat design. Figure 1 shows the hull of the FHsailbot and our test system Saudade. Table 1 compares the parameters of the boats.

The mold for the hull was shaped from a solid block of closed cell foam using an industrial robot equipped with a milling tool.

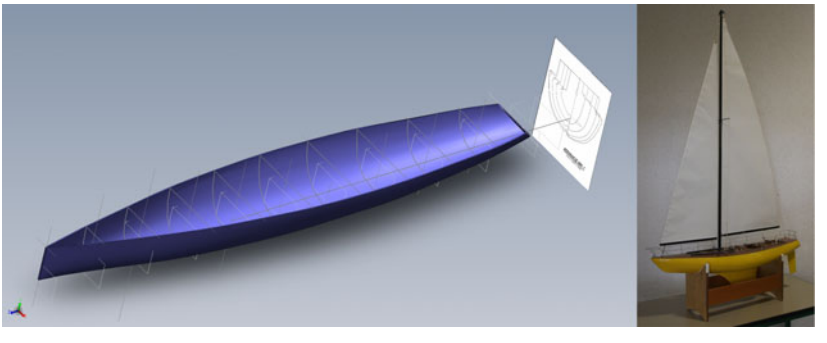


Fig. 1 The hull of the FHsailbot (left) and the test system for the control components (right)

[1] American Model Yachting Association.

Table 1 Basic parameters of the boats

Boat	FHsailbot	Saudade
Length overall	152 cm	112 cm
Length waterline	148 cm	103 cm
Beam	33 cm	26 cm
Draft	81 cm	26 cm
Displacement	∼15 kg	9 kg
Sail area	∼0.65 m^2	0.52 m^2

3 Control Hardware

At the beginning of the project we have a long discussion about the controller needed to control the vessel. Beside microcontrollers, we use microprocessor for mobile devices like notebooks in different multimedia projects (e.g. Pentium M (Intel, USA) and similar processors) and for small PC-like systems with 386/486-cores. On the other hand, we use microcontrollers from Texas Instruments (TI, USA) like the MSP430 for many small systems. As a compromise between performance and power needs, for this project, we decided to use an ARM7 microcontroller Stellaris Cortex-M3 from TI as the main controller. Only its use is discussed in this paper (see details later in this chapter). The controller gets data from different sensors (GPS, 3-axis compass, wind direction), generates PWM signals for sail winch and rudder servo and communicates over Bluetooth. An USB interface is used for programming and debugging. Status information may be displayed on a small OLED display. Beside the internal FLASH memory, additional data may be stored on a SD-card.

Figure 2 shows the system architecture of the control system. Beside the controller board with display and SD-card, the sensors (GPS, compass, wind direction) are providing the necessary environmental data, rudder servo and sail winch are directed by the controller. The failsafe module is able to overwrite the controller signals of rudder and winch using a remote control to avoid problems (e.g. collisions with other boats), if the controller does not work as intended.

3.1 Controller

All ARM7 microcontrollers of the Stellaris Cortex-M3 family from TI use the same core running at 50 MHz with different peripherals. Due to the three UART, we decided to use the LM3S6965 controller. If a separation of some peripherals from the main controller is necessary, e.g. due to limited wire length of some sensors, a controller with CAN bus may be used to solve the problem. Figure 3 shows a possible solution: LM3S8962 combined with the smaller LM3S2110. The sensor on the I2C port may be placed as far away as possible from metal components of a boat. The ports of the LM3S2110 may be controlled over the CAN bus from the LM3S8962.

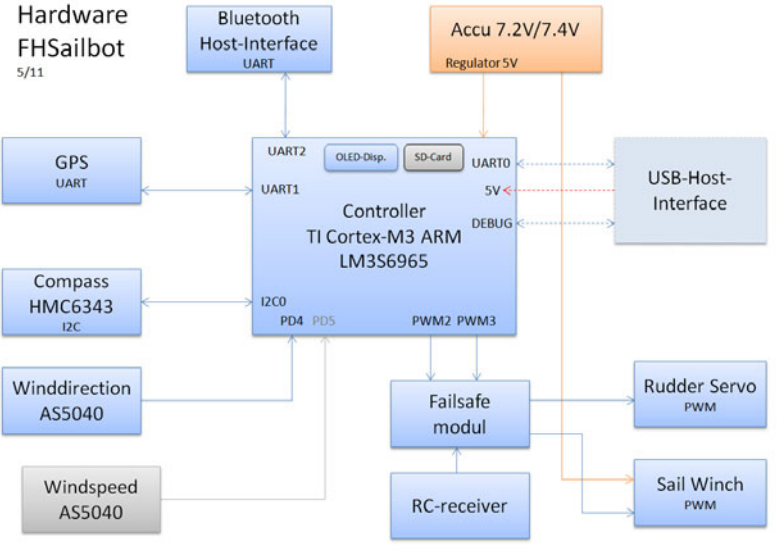


Fig. 2 System architecture of the control system: Sensors (left), actuators (right bottom), Bluetooth and USB communication and power supply

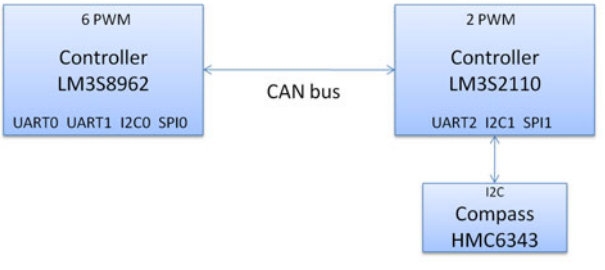


Fig. 3 System architecture with CAN bus, compass separated from the main controller

Table 2 shows the main characteristics of the controllers and the number of the different interfaces.

The controller needs a supply voltage of 3.3 V. All peripheral pins are 5 V-tolerant, that means, we dont need to change the logic levels of 5 V-peripherals. For the boat we use an evaluation kit (EK-LM3S6965) from TI, which provides the following additional components: In-circuit debugging interface, USB interface

Table 2 Main characteristics of the Stellaris Cortex-M3 controllers and the available interfaces

Controller	LM3S6965 (without CAN)	LM3S8962/LM3S2110 (CAN)
Main clock	50 MHz	50 MHz/25 MHz
Flash/RAM	256 KB/64 KB	
Timers	4x 32-bit or 8x 16-bit, SysTick, Watchdog	
I2C	2	1/1
SSI/SPI	1	1/1
UART	3	2/1
PWM	2x3	2x3/2
Ethernet	1	1
CAN	-	1/1

(communication over UART0, programming of the device, debugging), SD-card slot, buttons, status led and an 128x64 OLED display. Figure 4 shows the controller board and some other components.

3.2 Sensors

The boat gets data from three sensors: GPS, compass and wind direction, similar to [1]. The GPS is a small module from NAVILOCK (Germany) based on an ublox5 chip set (u-blox, Switzerland). Under free air conditions, the module is able to generate differential GPS data (Fig. 4b). With a free sight to the south, we try to

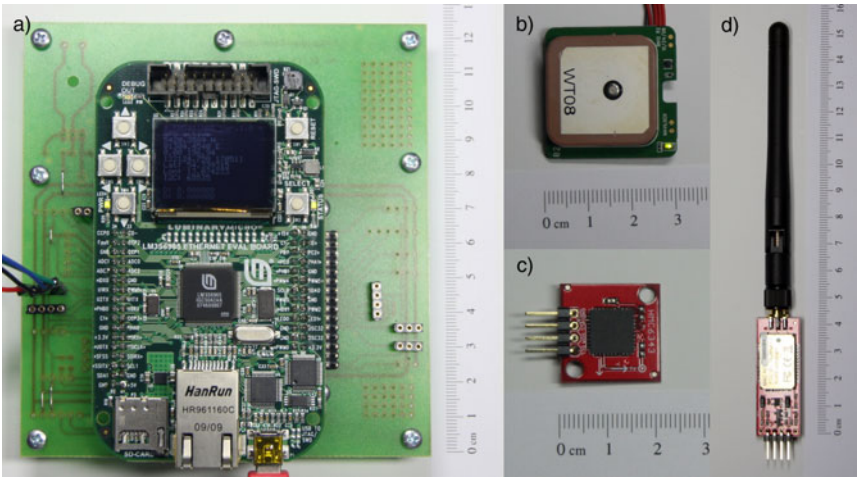


Fig. 4 Components: **a** Controller board EK-LM3S6965 with test adapter. **b** GPS. **c** compass. **d** Bluetooth modem.

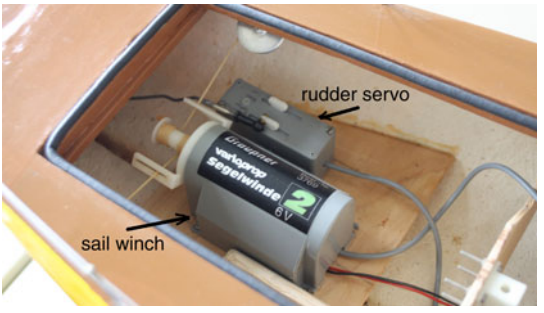


Fig. 5 Rudder servo and sail winch in the test system Saudade

receive the reports from the EGNOS[2] satellite ARTEMIS to increase the accuracy of the GPS. Unfortunately the connection to the EGNOS satellite is very unstable. The GPS module occupies one UART interface. As compass, we use the 3-axis compass HMC6343 (Honeywell, USA). It returns the tilt-compensated value for the heading and the heeling of the boat. It communicates over the I2C bus. To get the direction of the apparent wind, we use the PWM signal generated by the AS5040 (austriamicrosystems, Austria). To eliminate the tolerance of the PWM frequency, the controller gets the absolute value of the wind direction *w_dir* by the following calculation:

$$w\_dir = \left(\frac{w\_dir\_value \times 1025}{w\_dir\_period} - 1\right) \times \frac{360}{1024} \tag{1}$$

Where *w_dir* is the wind direction in degree, *w_dir_value* the active phase of the PWM signal and *w_dir_period* the whole period of the PWM signal.

3.3 Actuators

To control rudder and sail, we use standard components similar to the components used for the Saudade: a servo for the rudder and a sail winch with separate power supply to control the sails (Graupner, Germany). Figure 5 shows the components of the test system.

3.4 Power Consumption

For a small boat like our test system Saudade, the power consumption is a critical factor. To keep the system simple and cheap, we searched for a control system, which can be supplied by small NiMH oder LiPo (Lithium-polymer) accumulators. The chosen LM3S6965 needs less than 1 W. The power consumption and the cost of the components shows Table 3. Even if the computing power may not be sufficient for all algorithms, we think this is a good starting point.

[2] European Geostationary Navigation Overlay Service.

Table 3 Power consumption and cost of the control system components

Component	Current at Voltage	Cost
EKS-LM3S6965 at 50 MHz, display on	170 mA at 5 V	80 €
Bluetooth modem	30 mA at 5 V	54 €
GPS	80 mA at 5 V	50 €
Compass	5 mA at 3.3 V	130 €
Wind direction	20 mA 3.3 V or 5 V	
Rudder servo quiescent current	10 mA at 5 V	50 €
Rudder servo max. current	500 mA at 5 V	
Sail winch electronic	10 mA at 5 V	150 €
Sail winch motor max. current	3.5 A at 7.2 V	
Failsafe system incl. remote control receiver	30 mA at 5 V	50 €

The boat needs a minimum total current of about 360 mA and a maximum total current during the sail adjustment of about 4 A. To supply the necessary current, the boat is equipped with two accumulators: one for the control system and the sensors (regulated to 5 V) and another directly connected to the sail winch. The cost for the whole control system including the accumulators, sensors and actuators will be less than 1000 €.

3.5 Communication

Today there are two ways to communicate with the boat. During the development of the software, we use the USB interface to program and to debug the system and to exchange data using a virtual COM port on the host system. To get data from the boat and send data to the boat during sailing, we use a Bluetooth connection. The Bluetooth module connects to UART2 and transmits the serial data. The BlueSMiRF RP-SMA (Sparkfun, USA) connects to a standard 2.4 GHz antenna. On the host side, we use a PCMCIA-Bluetooth module with the same antenna. With this combination, we reach distances of about 100 m.

For a failsafe operation, the PWM signals of the controller may be overwritten using a standard remote control (RC). With a separate channel of the RC we can switch between the PWM signals of the RC and the controller signals.

Other communication modules will be included in future.

4 Software

The software of the control system is written in C. TI provides a driver library to encapsulate some of the hardware details, e.g. to program a timer or to communicate using an UART. To simplify the software development, there are different programming environments available. In this project, we use the IAR[3] Embedded

[3] IAR Systems, Sweden.

Workbench for ARM. An alternative are TIs Code Composer Studio development tools. Based on the in-circuit debugging interface of the evaluation kit, we are able to program and debug the controller from within the Embedded Workbench using the USB interface.

4.1 Operating System

Because of the different tasks the control system has to do during sailing (get the sensor data, calculate the course, generate the signals for actuators and communicate with the environment), we decided to use an operating system for the control system. There are different operating systems available for the ARM7 controller. We use the very small and stable operating system µC/OS-II. It has a small real-time kernel and has been certified within some commercial products to meet the requirements to be used in safety-critical systems. Details of the OS can be found in [2]. An alternative with compatibility to Linux is RoweBots Unison V5 RTOS (RoweBots, Canada).

4.1.1 Hardware Requirements

To use µC/OS-II as OS for a microcontroller, the controller needs to provide a system tick – a periodical interrupt (in this case with a period of 1 ms) as a basic clock for the scheduler. The LM3S6965 has a special timer to provide the system ticks.

4.1.2 Board Support Package

To hide more of the hardware details, a board support package (BSP) provided as a part of the port of µC/OS-II for the Cortex-M3 is used. The BSP may be extended corresponding to the systems requirements. Figure 6 shows the initialization of the AS5040 to get the wind direction. Commands with a leading *BSP_* are special commands of the µC/OS-II BSP, while the other commands are part of the driver lib.

4.1.3 Tasks

To create a task, µC/OS-II needs some initialization steps (see Fig. 7). After *BSP_Init*, *CPU_Init* and *OS_Init* the main task is created. In this task, all needed user tasks are created. Then the OS may be started (line 12). After this step the multitasking environment is running.

4.2 Interrupt Service Routine vs. Task

To show the different handling, e.g. to get sensor data, we compare the wind direction using the AS5040 with the heading using the compass (see Fig. 8). The wind direction is a PWM signal which needs to be analyzed by the controller. After the initialization, an interrupt service routine (ISR) is called whenever the logic level on the input pin changes. As result we got two values (see Fig. 9): The whole

```
01: // Timer Initialization
02: // Configure 16-bit periodic timer for wind direction
03: SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
04: TimerConfigure(TIMER0_BASE, TIMER_CFG_16_BIT_PAIR
                              | TIMER_CFG_A_PERIODIC);
05: // Set the prescaler to 50; Timer Tick is 1 MHz
06: TimerPrescaleSet(TIMER0_BASE, TIMER_A, 50);
07: // Enable the timer
08: TimerEnable(TIMER0_BASE, TIMER_A);
09: // Load start value
10: TimerLoadSet(TIMER0_BASE, TIMER_A, 0xFFFF);
11: // GPIO Port D Initialization
12: SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
13: // Use Pin 4 as input for the wind direction
14: GPIOPinTypeGPIOInput(GPIO_PORTD_BASE, GPIO_PIN_4);
15: // Generate an interrupt on both edges of the signal
16: GPIOIntTypeSet(GPIO_PORTD_BASE, GPIO_PIN_4,
                                    GPIO_BOTH_EDGES);
17: // Init the ISR for port D pin 4
18: BSP_IntVectSet(BSP_INT_ID_GPIOD, BSP_GPIOD_IntHandler);
19: // Enable interrupt
20: BSP_IntEn(BSP_INT_ID_GPIOD);
```

Fig. 6 Initialization of the port reading data from the wind direction sensor AS5040

```
01: BSP_Init(); // Initialize BSP.
02: CPU_Init(); // Initialize CPU.
03: OSInit();   // Initialize "uC/OS-II"
04: // Create the Semaphores
05: LCDSem = OSSemCreate(1); //LCD
06: UARTSem = OSSemCreate(1); //UART
07: // Create the start task
08: // The start task creates all other tasks
09: os_err = OSTaskCreateExt(...);
10: ...
11: // Start multitasking (give control to uC/OS-II)
12: OSStart();
```

Fig. 7 Initialization steps of µC/OS-II

period of the PWM signal (*W_Dir_Period*, line 21) and the time, the signal is '1' (*W_Dir_Value*, line 15). The wind direction variable is updated every 1025 µs (Fig. 8 right). This variable may be read by different tasks, which need the wind direction for internal calculations. Other tasks read sensor values over different interfaces when needed (Fig. 8 left).

Corresponding to (1) we can calculate the error-corrected value of the wind direction in degree. On the other hand, the compass HMC6343 is able to deliver

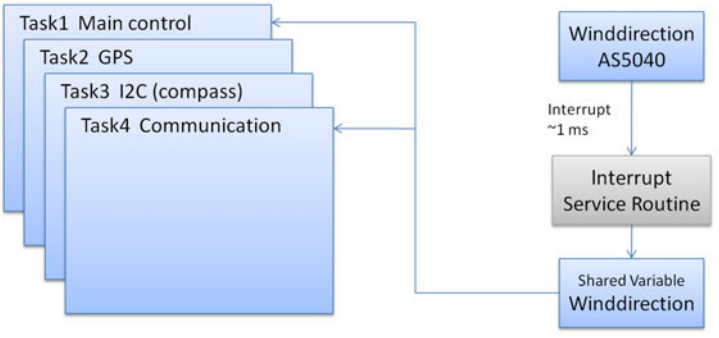


Fig. 8 Tasks reading sensor values (left) vs. interrupt driven calculations (right)

```
01: static  void  BSP_GPIOD_IntHandler (void)
02: {
03:   CPU_INT64U intstat;
04:   // Get GPIO interrupt source
05:   intstat = GPIOPinIntStatus(GPIO_PORTD_BASE, true);
06:   // Clear the GPIO interrupt.
07:   GPIOPinIntClear(GPIO_PORTD_BASE, GPIO_PIN_4);
08:   // Interrupt on Pin 4 (Wind direction)
09:   if (intstat & GPIO_PIN_4)
10:     if (GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_4) == 0)
11:     {
12:       // Falling edge
13:       // Save the time since the rising edge
14:       W_Dir_Value =
               0xFFFF-TimerValueGet(TIMER0_BASE, TIMER_A);
15:     } else {
16:       // Rising edge
17:       // Save the time since the last rising edge
18:       W_Dir_Period =
               0xFFFF-TimerValueGet(TIMER0_BASE, TIMER_A);
19:       // Reload the timer
20:       TimerLoadSet(TIMER0_BASE, TIMER_A, 0xFFFF);
21:     }
22: }
```

Fig. 9 Interrupt handler for port D. Detect changes on BIT4 (PWM signal of the AS5040)

the heading directly over the I2C bus with a default rate of 5 Hz. So we can get new values every 200 ms. Figure 10 shows the code for a task reading the data of the HMC6343 once a second. *OSTimeDly* ensures that the task sleeps for 1 s (see line 13). The tasks may be prioritized depending on the requirements. The

```
01: static  void  App_TaskI2C (void *p_arg)
02: {
03:   CPU_INT08U err;
04:   CPU_INT16U heading;
05:
06:   while (DEF_TRUE) {
07:     // get the heading of the boat and display it
08:     heading = BSP_I2C_Read_HMC6343();
09:     OSSemPend(LCDSem, 0, &err);
10:     sprintf(App_LCDLine9, "I2C: %d       ", heading);
11:     OSSemPost(LCDSem);
12:
13:     OSTimeDly(OS_TICKS_PER_SEC);
14:   }
15: }
```

Fig. 10 Task reading the compass data over the I2C bus once per second

combination *OSSemPend*/*OSSemPost* locks the display for this task to ensure a correct viewing of the value (line 9/11). The exclusive access to a shared variable is possible in the same way. Shared components in this system are the display and the Bluetooth UART. They are locked with semaphores.

4.3 Control Software

For our first tests, we used a modified fuzzy control algorithm based on [3]. Currently, only the low-level functions for capturing the sensor data and controlling the actuators are embedded software modules running on the LM3S6965, whereas all high-level software components are implemented on a laptop computer connected via Bluetooth. The laptop gets the sensor data and sends commands back to the boat. Depending of the distance, the laptop is also used as remote control. In the final set-up, all software components are running on the LM3S6965, the laptop is only used to show the status of the boat.

5 Conclusion

We have shown our approach building an autonomous sailboat. In June, the hull of the boat is laminated. The low-level software components are ready, the high-level software components are work in progress. The work on both parts (the boat and the controlling software) gets on, but we will still have to do many tests and improvements. Our goal is the participation of our FHsailbot at the WRSC2011 in Lübeck.

References

1. Ammann, N., Hartmann, F., Jauer, P., Bruder, R., Schlaefer, A.: Design of a robotic sailboat for WRSC/SailBot. In: IRSC 2010, Canada (2010)
2. Labrosse, J.J.: MicroC/OS-II, The Real-Time Kernel. CMP Books, San Francisco (2002)
3. Stelzer, R., Pröll, T., John, R.: Fuzzy logic control system for autonomous sailboats. In: IEEE International Conference on Fuzzy Systems (2007)

Simulating Sailing Robots

Colin Sauzé and Mark Neal

Abstract. This paper outlines our experiences in simulating sailing robots. It focuses on efforts to simulate a sailing robot, both in pure software and a "Hardware in the Loop" (HIL) simulation and compares these results with sailing a similar course using an actual robot. The software simulator is built upon an open source sailing game called Tracksail. This is based on a highly simplistic physics model and makes no attempt to simulate the actions of waves, currents or tides. The target robot to be simulated is BeagleB, a 3.65 m long boat based upon a MiniJ dinghy equipped with a 2m high wing sail. To provide a more realistic simulation of this boat a "Hardware in the Loop" simulator was constructed using nearly identical electronics attached to a trolley table. Sensor inputs can either be provided by a Tracksail simulation or from sensors on the table. Sailing of an identical course in the Tracksail, the HIL System and on BeagleB showed similar results in all cases. This suggests these simulations are a reasonable approximation of real world robot performance. Although they are no substitute for actually sailing a robot they are a sufficient for testing software, testing hardware and developing control strategies.

1 Introduction

Simulators offer an opportunity to develop and test sailing robot control algorithms without incurring the overheads of deploying, operating and recovering a real robot. These overheads can significantly slow development of control systems hardware and software and can create difficulties in comprehensively testing all aspects of a control system. A simulator offers a mechanism to overcome these limitations and produce better tested control systems. However, simulators also suffer from a number of key disadvantages, it is difficult to create a realistic model of a boat's motion through water (especially one which can be executed in real time) and to model the detailed effects of waves, tides, winds shifts and gusts.

Colin Sauzé · Mark Neal
Department of Computer Science, Aberystwyth University
e-mail: {cos,mjn}@aber.ac.uk

This paper outlines our experiences in simulating sailing robots. It focuses on efforts to produce software only and hardware in the loop (HIL) simulators. These results are compared against sailing similar voyages on real robots.

2 Background

In previous work [9] a simulator based upon an open source sailing game called Tracksail [6]. An API was implemented by sending commands over a TCP/IP socket between a client application and Tracksail. This interface followed a mechanism similar to that which we used in some of our early sailing robots. This is based on a highly simplistic physics model and makes no attempt to simulate the actions of waves, currents or tides. It allows variable wind directions through a simplistic mechanism which specifies upper and lower bounds for the wind direction and the extent to which it will vary.

We have observed a number of key differences in behaviour between Tracksail and real sailing robots. In particular the response rates to rudder and sail movements of the simulated boat are much faster than real boats and there is no lag between initiating an actuator movement and it completing. Despite these limitations, in previous work using Tracksail to develop power management strategies similar trends were exhibited between Tracksail and real sailing robots [10].

Another possible approach for simulation might be to adapt commercial games such as Virtual Sailor [2] or Sail Simulator [1]. These both offer simulations of a variety of boats, waves, varying wind conditions and appear to have far more realistic physics models than Tracksail. However, being closed source games they are not intended for sailing robot simulation and may not easily be adaptable for this purpose. One possible means to achieve this might be to communicate with them through the network protocols intended to provide multi-player support.

There is relatively little other scientific literature regarding the simulation of sailing robots. Briere (2008)[4] produced a MatLAB simulation of a 2.4 m long sailing robot. This simulation included both consideration of the aerodynamic and hydrodynamic aspects of the boat. It also presented an software interface similar to that used by the actual boat in order to allow for development of a common code base. He presents results showing a close match between sailing the same course in the real world and in simulation. Mathematical models of sailing boat movement ([8, 7] have also been developed, although some effort will be required to bring these to the point where they can form part of a real time simulation environment.

3 Methods

This section discusses the BeagleB sailing robot, the Tracksail simulator and an HIL simulator, which combines Tracksail with a copy of BeagleB's electronics.

3.1 BeagleB Sailing Robot

3.1.1 Control System Overview

BeagleB is a 3.65 m long MiniJ sailing dinghy, its full specifications are shown in Table 1. Autonomous control is provided by a Gumstix single board computer running a Linux based operating system. The Gumstix processes data from the wind sensor, GPS and compass and determines appropriate actuator positions to sail its target course. Low level control of actuator positions is performed by a PIC18F4550 microcontroller which receives target positions from the Gumstix. A separate system of 4 LEM CAS 6-NP hall effect current transducers are used to monitor the power consumption of the entire system. One transducer is attached to each actuator, another to the output of the solar panels and a fourth to the battery output. These produce a voltage proportional to the current passing through them, these voltages are processed by an Arduino Uno (ATMega328) microcontroller which is able to pass this data on via an ethernet interface to the Gumstix in the form of a UDP packet sent every 10 seconds. This process allows for power consumption to be tracked over time.

3.1.2 Long Term Power Management

BeagleB is intended for the dual purpose of performing autonomous oceanographic monitoring and researching long term power management in sailing robots. Sailing robots offer a unique opportunity to for relatively low cost and low risk research power management strategies for robots which must operate without contact and supervision from human operators for prolonged periods of time. The relatively small size of BeagleB (and the general desire to minimise size, cost and potential danger to other vessels) limits the amount of power which can be generated by onboard

Table 1 The specifications of BeagleB

Name	BeagleB
Hull	3.6 5m long Mini-J dinghy
Keel	80 kg single keel
Sail	2 m high carbon fibre wing sail
Sail Actuator	LINAK LA-12
Rudder Actuator	LINAK LA-12
Computers	PIC18LF4550 and Gumstix
Wind Sensor	Furuno Rowind
Batteries	2.8 kW h sealed lead acid batteries
Solar Panels	2 panels, 90 W peak total
Compass	Furuno PG-500 Fluxgate Compass
GPS	Furuno GP-320B
Power Tracking	4 LEM CAS 6-NP Transducers sampled by an ATMega328 Microcontroller

photovoltaic solar panels and leaves a power budget which is only just sustainable for long term operation (at moderate latitudes, winter time operation at extreme latitudes would still not be possible). Given these limitations , power must be managed efficiently. To achieve this a biologically inspired mechanism based upon the mammalian endocrine system has been implemented [10, 11]. The endocrine system operates by secreting chemical messengers called hormones into the blood stream [5]. These spread throughout the body in a broadcast fashion, however they only effect certain target cells. When they encounter a target cell the hormone will bind with receptors on the surface which are shaped only to fit that particular hormone. After this binding occurs the hormone will trigger a behavioural change within the internal workings of the target cells. Hormones are responsible for the regulation of a number of biological processes including sleeping patterns, blood sugar levels, salt levels, blood pressure, coordination of actions between the neural, immune

Fig. 1 A Photograph of BeagleB during the 2009 World Robotic Sailing Championships in Matosinhos, Portugal

and endocrine system and triggering the "fight or flight" response which occurs in dangerous situations. These regulatory properties combine together to form a stable internal state within the body known as homeostasis. Creating an artificial homeostasis within a sailing robot would enable it to modify its behaviour in accordance to varying conditions. For example, taking the analogy of the hormone insulin which controls the uptake of glucose from the bloodstream. An artificial insulin could be used to activate power consuming behaviours such as switching on oceanographic sensors, using communications equipment or making more frequent or larger course adjustments when battery levels are plentiful. When battery levels are low these behaviours can be suppressed to reduce power consumption. The robot's activities could also be promoted and suppressed in response to diurnal or seasonal solar cycles or tides so that power consuming activities are coordinated with the presence of favourable conditions.

Getting BeagleB to achieve the long missions required for this research is a non-trivial task. Due to the financial cost of the robot and less than perfect reliability record of the onboard electronics and software, we are currently unwilling to allow it to operate for prolonged periods without supervision from a chase boat or to operate it in heavy seas. Achieving this time commitment (especially when combined with the overheads of launching and recovering BeagleB) has been difficult and has severely limited the amount of time for on the water experiments. Simulations offer the ability to run many iterations of these experiments without the need for chase boats and chase boat crews. They also offer a means to debug and perfect software and hardware before they are deployed at sea in a more challenging environment where the cost of failure is far higher.

3.2 Tracksail-AI Simulator

As discussed in section 2, in previous work we have modified the Tracksail open source sailing game to produce a sailing robot simulator. This derivative known as Tracksail-AI [3] allows a client program to connect to the simulator using a TCP/IP socket. It can then issue commands which allow it to control the rudder and sail positions, read the wind direction, boat heading and location. The Tracksail-AI server is written in Java and presents a graphical interface illustrating the wind direction and the boat's heading, location, rudder position and sail settings. Additional client side code is used to calculate the state of a simulated solar panel (based upon time of day, time of year and location) and to track the battery state based upon the movement of actuators and which sensors are considered to be switched on (up to date information will not be made available for any sensor which is considered to be off). Through this power consumption data can be generated and experiments with artificial endocrine systems (as discussed in the previous section) undertaken. In our latest version of Tracksail there has been a focus upon parallelisation with the ability to run multiple instances of the same experiment in parallel. Jobs can also be distributed across multiple computers in order to carry out simulations with a wide

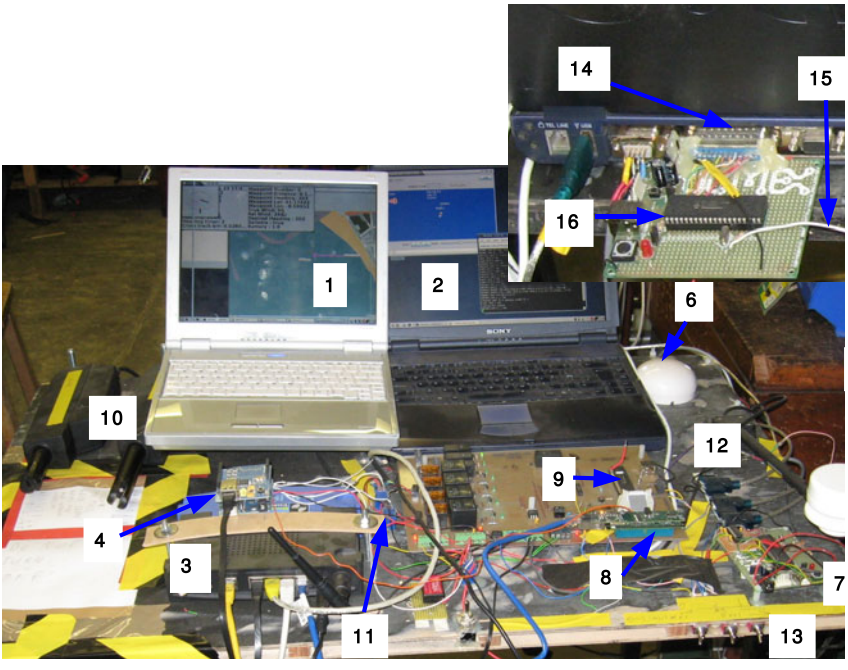


Fig. 2 A photograph showing the electronics used in the Hardware in the Loop simulator. Numbered labels: 1 - Telemetry Laptop, 2 - Tracksail Laptop, 3 - Wifi Access Point, 4 - Arduino, 5 - Wind Sensor, 6 - GPS Antenna, 7 - Compass, 8 - Gumstix, 9 - PIC, 10 - Acuator, 11 - Current Transducers, 12 - USB to Serial converters, 13 - Data Source Switches, 14 - Parallel Port Connector, 15 - Cable from Solar Simulation PIC to Arduino, 16 - Solar Simulation PIC.

range of experimental parameters. It is hoped that this can help reduce the size of the parameter space that needs to be run upon BeagleB and the HIL simulator.

3.3 Hardware in the Loop Simulation

In order to provide a more realistic simulation an HIL system, based upon BeagleB was created. This used (almost) identical hardware to BeagleB placed on a trolley table which can be easily moved around. A photograph of this system is shown in Figure 2. An alternative configuration uses a laptop computer running Tracksail to provide mock GPS, compass and wind sensor inputs. This setup was primarily intended as a debugging aid, unlike a standalone Tracksail simulation this allowed code identical to that which runs on BeagleB to be run on the simulator. The ability to run long term simulations with an identical code base provides a valuable tool for testing sailing robot control systems for bugs. Without this approach much of the time spent deploying BeagleB would need to be spent debugging low level code which differs from the Tracksail version rather than actually running scientific experiments.

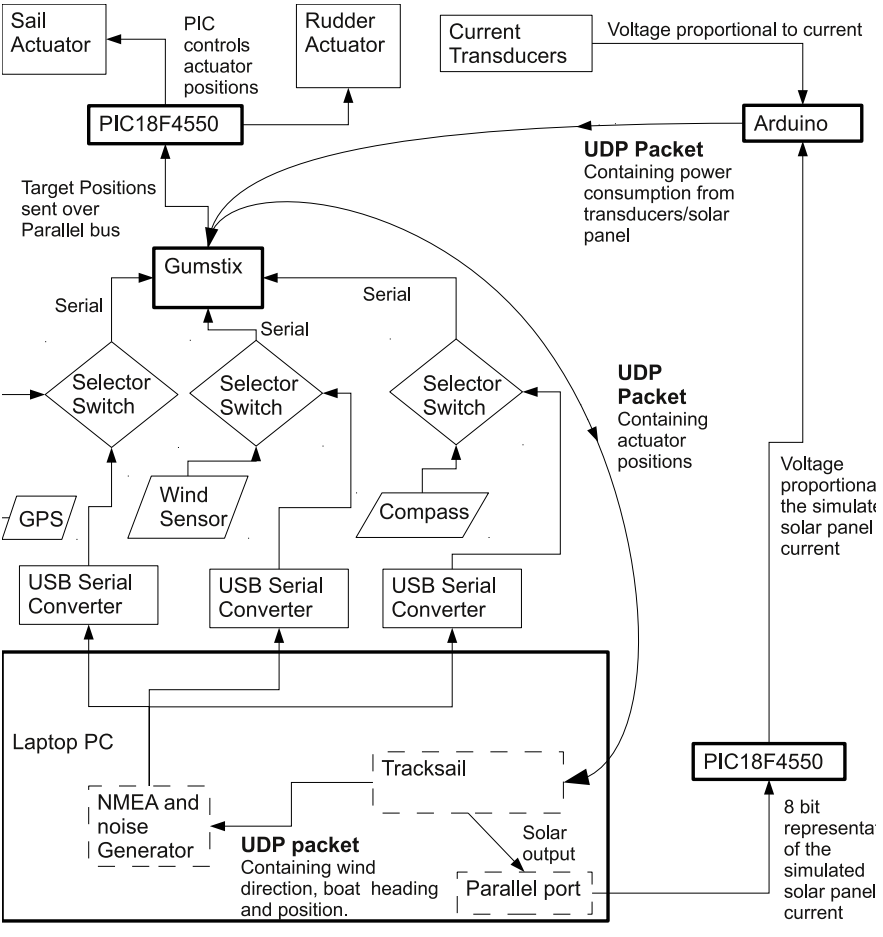


Fig. 3 A figure showing the dataflows used by the Hardware in the Loop Simulation. Compass, GPS and Wind data can either be generated by real sensors or from the laptop running Tracksail. The three selector switches determine which inputs are used by the Gumstix. The Gumstix calculates actuator positions, depending upon the wind, GPS and compass data. These positions are sent to the PIC which positions the actuators. The Gumstix generates UDP telemetry packets, which include actuator position data. This is fed back to Tracksail running on the laptop so that it can change its actuator positions. The Gumstix also receives data on power consumption from the Arduino, which processes data from the current transducers. As there is no real solar panel on this system, its current transducer output is simulated by a PIC controlled via the parallel port of the laptop.

Figure 3 shows the data flows for the process. Tracksail generates a UDP packet every-time it completes its main loop which contains the simulated boat's heading and position and the wind direction. These are then received by another process running on the laptop which applies noise to them and generates data in the NMEA 0183 format used by the real sensors. The noise is created using a Gaussian distribution, with wind and compass data being subjected to noise ranging between +/- 2 degrees and GPS data ranging between +/- 5 metres. To match the real sensors GPS and wind data are limited to transmission at 1hz and 4800 bits per second and compass data at 4hz and 19200 bits per second. This sensor data is then received by the Gumstix and processed by the same code as run's on BeagleB. To complete the loop the Gumstix then transmits a UDP packet (normally used to send telemetry data back to a monitoring station) which contains the actuator positions. This packet is received by Tracksail and used to set its actuator positions. There is a slight loss of realism at this stage as this packet contains the target position of the actuator rather than its actual position. This is due to the limitation that the Gumstix is not actually aware of the actual actuator position and can only send target positions to the PIC as part of a unidirectional data transfer.

This configuration was also constructed with the aim of accurately simulating and monitoring the power consumption of the robot. To achieve this (as with BeagleB) a set of hall effect current transducers are used to measure the current drawn by the actuators and other electronics. These are connected to an Arduino microcontroller in the same manner as BeagleB, except there is no current transducer from a solar panel. Instead, the solar panel current is calculated by Tracksail based upon the sun elevation for the current time/date and location. This is the represented as an 8 bit value (0-255) and is transmitted over the parallel port to a PIC18F4550 microcontroller. The PIC converts it to a PWM signal, which is in turn converted to an analogue voltage (0-5V) using a filter circuit built from a resistor and capacitor. This analogue line then connects to the Arduino, replacing the input from the solar panel current transducer.

4 Experiments and Results

A series of experiments were carried out to test the difference between a pure Tracksail environment, the HIL simulator and BeagleB. A triangular course of approximately 1.2km was selected. This was based upon a race undertaken on July 11th 2009 during the 2009 WRSC (World Robotic Sailing Championships) in Matosinhos, Portugal. The course was sailed in both simulated environments and compared with the actual data from BeagleB during the 2009 WRSC. In all cases the wind was blowing from the North West at approximately a Beaufort force 2. Table 2 compares the time taken, actual distance covered and number of actuator movements made by each. Figure 4 compares the courses of all three.

From Figure 4 it can be seen that the courses are not quite identical. There is a slight offset between them caused by different co-ordinate systems used by Tracksail and the hardware systems, the starting point also varied slightly due to an initial

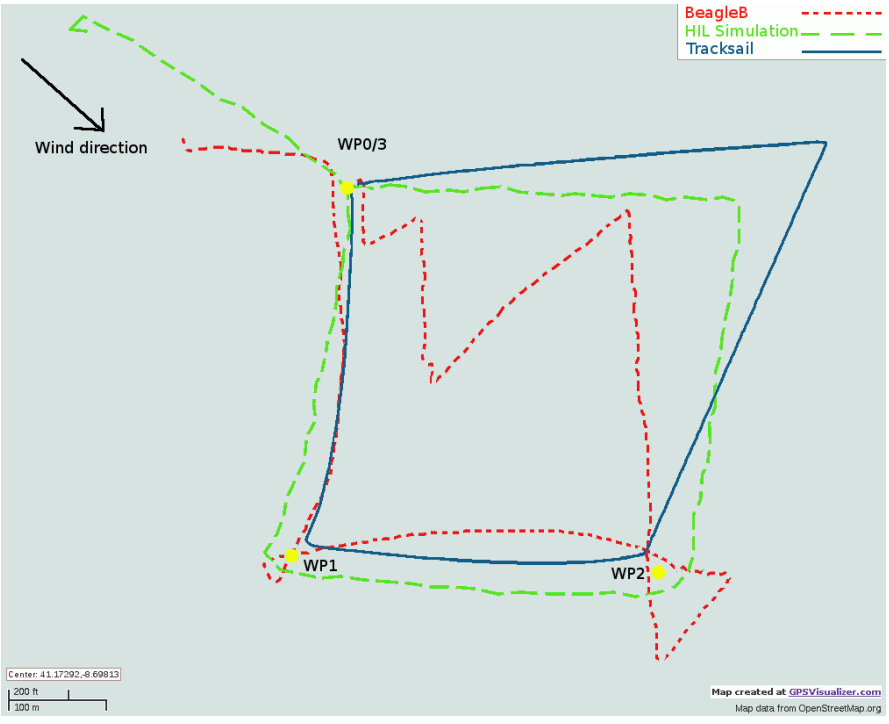


Fig. 4 Map showing the GPS tracks.

Table 2 A comparison between the simulator, HIL simulation and BeagleB

Type	Distance Sailed	Time Taken	Rudder Movements	Sail Movements
Simulator	1.69 km	7403 seconds	196	33
HIL Simulation	1.98 km	902 seconds	138	528
BeagleB	2.32 km	5168 seconds	267	96

drift while the control system was starting. A bug in BeagleB's code caused it to struggle to turn more than 127 degrees, this caused problems rounding waypoint 1 and waypoint 2. During the upwind leg between waypoints 2 and 3, BeagleB tacked several times, this is despite the tacking algorithm being used, which attempted to make only one tack. This was due to noise from the wind sensor causing the control system to believe it was now able to sail directly to the waypoint, but after tacking the wind direction would shift.The lower levels of noise in the HIL simulation and the Tracksail simulation, did not cause this behaviour to be exhibited. The angle at which the boat can sail to the wind is also different in each system (hence the different angles on the upwind leg). BeagleB was set to attempt to sail at 35 degrees to the wind, while the HIL system requires at least 45 degrees due to limiations

in Tracksail. The Tracksail simulation was actually set to 55 degrees due to code inherited from another robot, which could only sail at 55 degrees.

The total time taken by the Tracksail simulation and the HIL simulation seem to differ massively, this may be due to the way time is calculated. The Tracksail simulation considers 1 second to have elapsed after each iteration of the loop, while the HIL simulation uses a realtime clock to calculate this. The number of sail movements in the HIL simulation is also much higher than either of the other methods. This is mainly due to a period when the sail position oscillated between two positions (the control system only has 11 possible positions it can place the sail actuator into) for several minutes. This problem has been observed in both BeagleB and Tracksail, but on this occasion only happened to occur with the HIL simulation.

Despite these differences, the performance between all three systems is broadly similar, with differences being within an order of magnitude apart. This suggests that using either Tracksail or the HIL simulation are likely to see similar results to those of BeagleB. When using these methods to verify algorithms and experiments to run upon BeagleB they should therefore be sufficient to at least eliminate those which will not be viable.

Power management results are not available at this stage as the version of the experiment on BeagleB was performed before power logging hardware was installed. The Arduino and current transducer system has also been problematic in initial development due to random reboots of the Arduino under high current loads and accuracy issues with the current transducer under low loads. A crude comparison of power useage can be made by comparing the number of actuator movements. Given that these are of similar orders of magnitude, this indicates that there is a reasonable match between the simulator, HIL and BeagleB in terms of actuator use and actuator power consumption. However, these values do not give a complete picture as they do not distinguish between large and small movements. Further data is required before we can conclusively demonstrate that either Tracksail or the HIL system are reasonable approximations for the power consumption of BeagleB.

5 Conclusions

Through this work we have shown that simulators can be useful for testing and development of sailing robot control algorithms. Even simulators with limited realism and simplistic physics models such as Tracksail can provide a valuable aid to the process of control system development and debugging and to act as a parameter verification method for scientific experiments investigating power management. However, even with an improved physics model no simulator is likely to be a full substitute for testing in a real environment. Capturing the required level of detail to represent the sheer complexity and dynamic nature of the world encountered by a sailing robot would be a monumental task. Given this simulation cannot provide an alternative to deploying a sailing robot in the real world, but it maybe useful to the development process. The use of “Hardware in the Loop” simulation provides a further aid to software and electronics development by offering a identical or almost

identical platform to that which will be deployed at sea. It can create an environment in which it is easy to instrument the system to observe electrical characteristics or software performance.

5.1 Future Work

Future work needs to focus on further improvements to the simulation model. It may be possible to bring in some of the models proposed by Roncin or Riddler, Vermeulen and Keuning [8, 7] or to make use of more realistic commercial systems. Tracksail's simulation is based around a traditional fabric sail, rather than an actuator controlled wing sail. This causes a number of behavioural differences compared with BeagelB, for example the sail will automatically jibe as the stern turns through the wind, while the wing sail must be actively turned to achieve this. The simulator could be adjusted to match the behaviour of a wing sail to give more accurate performance. Simulations of tides and currents could be introduced. Wind data could be based upon either real time feeds from an outdoor weather station, generated upon average data from internet sites or replayed from previously recorded data. Another possibility is to produce large lookup tables based upon actual performance data of a real sailing robot operating at sea. Hardware in the Loop simulations could be made more realistic by adding weights to the actuators to simulate the presence of loads against them. It might even be possible to run Hardware in the Loop Simulations on a full sailing robot while it is moored. This would result in the robot experiencing the friction of moving actuators and moving them against the wind and water. Further work needs to be undertaken to complete the production of a reliable power monitoring system and to compare the results of power consumption estimates from Tracksail and actual recorded data from the HIL simulator and Tracksail. The level of noise in the HIL system also needs to be varied to see if it is possible to recreate results seen on BeagleB (e.g. the noise which caused multiple upwind tacks to be required in figure 4). Noise could also be applied to the simulated sensors in Tracksail and this might give an increased level of realism to experiments run within Tracksail.

Acknowledgements. This research was funded by EADS Foundation Wales as part of the grant 'Tethys'.

References

1. Sail simulator, `http://www.sailsimulator.com/` (accessed June 10, 2011)
2. Virtual sailor, `http://www.hangsim.com/vs` (accessed June 10, 2011)
3. Tracksail-ai (2009), `http://sourceforge.net/projects/microtransat/files/tracksail-AI.tar.gz/download` (accessed June 10, 2011)
4. Briere, Y.: Iboat: An autonomous robot for long-term offshore operation. In: Proceedings of MELECON 2008, pp. 323–329 (2008)
5. Crapo, L.: Hormones: The Messengers of Life. W. H. Freemany and Company, New York (1985); ISBN 0-7167-1753-0

6. Kuusela, T., Brockmann, S., Johnson, R.: Tracksail (2004), http://tracksail.sourceforge.net (accessed June 10, 2011)
7. Ridder, E.J., Vermeulen, K.J., Keuning, J.A.: A mathematical model for the tacking maneuver of a sailing yacht. In: Proceedings of the International H1SWA Symposium on Yacht Design and Yacht Construction (2004)
8. Roncin, K., Kobus, J.M.: Dynamic simulation of two sailing boats in match racing. Sports Engineering 7(3), 139–152 (2004)
9. Sauzé, C.: Control software for a sailing robot. Master's thesis, University of Wales, Aberystwyth (2005)
10. Sauzé, C., Neal, M.: A neuro-endocrine inspired approach to long term energy autonomy in sailing robots. In: Proceedings of TAROS (Towards Autonomous Robotic Systems) (2010)
11. Sauzé, C., Neal, M.: Long term power management in sailing robots. In: Proceedings of IEEE Oceans 2011 (2011)

Part IV
Collision Avoidance

Automatic Obstacle Detection for USV's Navigation Using Vision Sensors

Oren Gal

Abstract. This paper presents an automatic method to acquire, identify, and track obstacles from an Unmanned Surface Vehicle (USV) location in marine environments using 2D Commercial Of The Shelf (COTS) video sensors, and analyzing video streams as input. The guiding line of this research is to develop real-time automatic identification and tracking abilities in marine environment with COTS sensors. The output of this algorithm provides obstacle's location in x-y coordinates. The ability to recognize and identify obstacles becomes more essential for USV's autonomous capabilities, such as obstacle avoidance, decision modules, and other Artificial Intelligence (AI) abilities using low cost sensors. Our algorithm is not based on obstacles characterization. Algorithm performances were tested in various scenarios with real-time USV's video streams, indicating that the algorithm can be used for real-time applications with high success rate and fast time computation.

1 Introduction

One of the most difficult challenges for USV navigation is to recognize and identify obstacles around the vehicle without human intervention. This task is known as Automatic Target Detection (ATD). An efficient ATD system should achieve high detection percentage for targets while maintaining a minimal false-alarm rate. This means that it must preserve an optimal balance between a high detection rate and a low error probability.

Although, ATD algorithms are very sensitive and unstable regarding clutter elements, i.e. elements that are not targets but still part of the scenes with similar characteristics as the targets. Dealing with clutter in ATD algorithms was extensively studied [1, 2, 3].

Oren Gal
Technion, Israel Institute of Technology
e-mail: orengal@tx.technion.ac.il

One of the ATD algorithms methods is based on the target temperature. The contrast of the target were based on environment gradient of the target and the environment's contrast to recognize targets. These methods suffer from false alarms due to targets and environment similarity. Several methods were developed to characterize the targets and to distinguish between the target's and the environment's characteristic, avoiding false alarms [3, 4].

Heuristic methods were introduced in the early 1980s based on threshold gradient in the image. The threshold was determined by the contrast of an object with its local background [3]. The segmentation part of such algorithms is based on standard edge operator using closing shape algorithms and filling steps [5, 6, 7, 8].

Most of the previous works were studied in aerial and ground environments without considering special phenomenas in marine environments, such as waves, clutter, vehicle's stability, etc. For the first time, our algorithm deals with ATD algorithms for USV's motion planning and automatic decision models using COTS sensors. The algorithm based on common video format as input, and computes targets locations around the vehicle.

2 General Description

Our new algorithm gets an input video in one of the common formats (avi, mpeg, mov, etc.), starts a few processing steps and eventually finds all the targets in the specific image.

The main steps as shown in Figure 1 are:

1. Read input image from the COTS device
2. Resize and convert the image into gray-scale reducing CPU time
3. Reduce search space for targets by preforming initial cleaning and horizon identification
4. Learn the sea pattern using a co-occurrence matrix
5. Morphologic cleaning, which cleans the image after distinguishing between sea and possible targets
6. Skeletonize the recognized targets in order to characterize them with a simple structure
7. For each target: Lower the characteristic structure into the "interesting points" according to the target skeleton structure that was found

Basic methods in image processing enable to indicate targets position at a specific frame with a small effort. Naive algorithms report target's current parameters immediately. Instead, we analyze the 20^{th} input frame at each time to find possible targets. We count the input frames and treat only the current 20^{th} input frame, comparing and correlating the results with the last 20^{th} input frame .We decided to use the 20^{th} input frame as optimal frame number based on the parameters of minimal

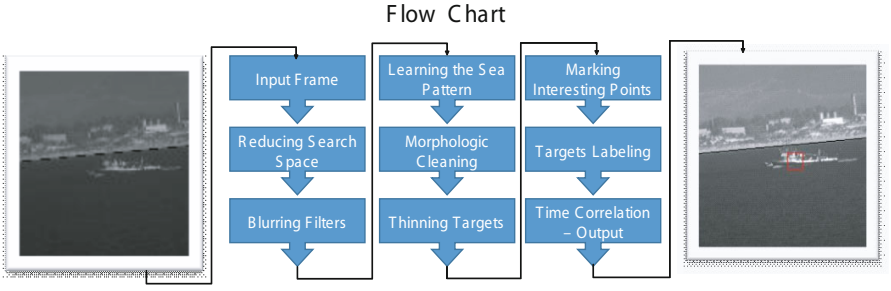


Fig. 1 Flow Chart of the Algorithm: Describing algorithm's functions from input frame to algorithm output using major steps: Reduce search space, Learning sea pattern, Morphologic cleaning and Skeletone algorithm

target changing in marine environments and CPU time computation. We analyzed these parameters on our real-time video movies with several frame numbers and decided to use the 20^{th} input frame.

By that, the algorithm ignores waves and clutter interferes and decreases the rate of false alarms [8, 11, 12]. Later on, we define the Interesting Points concept. The Interesting Points, which survived the time correlation are eventually defined as targets. Multiple targets can be found in each frame and each identified target is reported only once.

3 The Algorithm

3.1 Initial Cleaning

The marine environment suffers from variety of noises of different kind. In the first step, we apply a number of image blurring techniques to get a smoother texture of the sea. The input image might contain many noised pixels that will interfere with reducing the search space. This may become critical for real-time performances.

Therefore, a number of Gaussian softening filters are used. In order to preserve the horizontal structure of the sea pattern, we use convolution with a specific matrix designed to preserve the horizontal structure.

The initial cleaning part includes the Adaptive Smooth filter. As well known in the image processing world, this filter is designed to blur the image and save the sharp edges. This characteristic becomes very crucial in later stages to recognize and identify the targets. The Adaptive Smooth filter $w(x,y)$ is a very efficient filter, but hard to deal with "salt and pepper" clutter. The filter computes as:

$$w(x,y) = e^{-\frac{G_x^2+G_y^2}{2\cdot fac^2}} \tag{1}$$

where:

$$G_x(x,y) = \frac{l(x+1,y) - l(x-1,y)}{2} \tag{2}$$

$$G_y(x,y) = \frac{l(x,y+1) - l(x,y-1)}{2} \tag{3}$$

where fac in Eq. 1 sets the smoothing quality.

The Adaptive Smooth filter contains two Gaussian matrixes: Cleaning and Blurring matrix calculated by Eq. 1. The Cleaning Matrix, $C[i,j]$ is an $N \times N$ matrix (in our case $N = 7$), the specific values that were found to be the most efficient in the simulations are detailed below:

$$C[i,j] = \begin{pmatrix} 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \end{pmatrix} \tag{4}$$

The Blurring Matrix $G[i,j]$ is a $N \times N$ matrix, where $N = 6$, the matrix blurs the image and is known as a very effective one dealing with noises:

$$B[i,j] = \begin{pmatrix} 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \\ 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \\ 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \\ 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \\ 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \end{pmatrix} \tag{5}$$

3.2 Reducing the Search Space

Motion planning and autonomous decision models are often to be done in real-time. For that, the algorithm must consider computation time constraints and optimize the CPU time at each time step.

Therefore, the initial step is to recognize the "interesting" parts in the frame; in our case, the part that contains sea texture. We characterize the sea texture by defining the horizon line l and the horizon line orientation θ. The identification is computed using the Canny Edge Detector algorithm computing the gradient intensity at each point (x,y) in the frame:

$$l(x,y) = \sqrt{\nabla x^2 + \nabla y^2} \tag{6}$$

$$\theta = tan^{-}1\frac{\nabla y}{\nabla x} \tag{7}$$

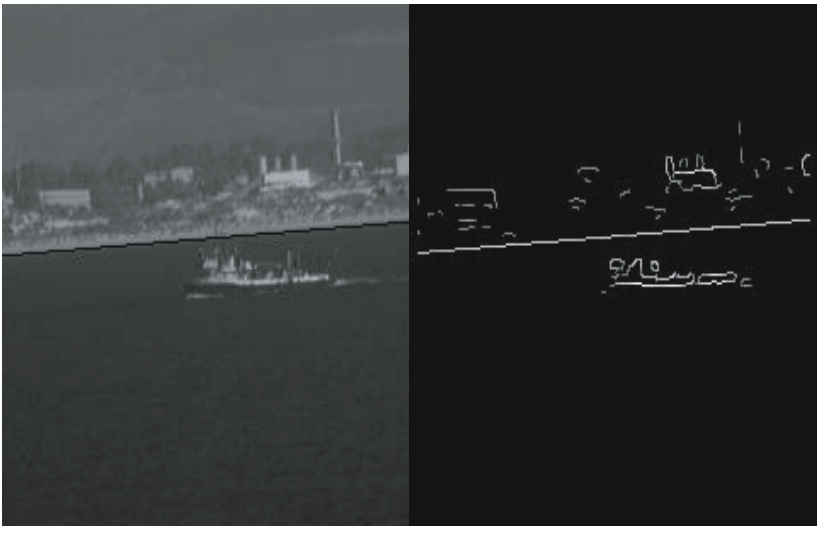


Fig. 2 Initial Cleaning of the Image. The left side shows the original input image after the transformation into a gray scale format. The right side shows the output image of the Canny Edge Detector.

Figure 2 shows the original input and the output image of the Canny Edge Detector. The next step is to reduce the search space using a simple Hough Line Transformation to get the most intense line in the image.

3.3 Learning Sea Pattern

This part is the actual recognition part of the algorithm. We assume that there are no targets within 10 pixels from each edge of the frame. This assumption is legitimate, based on the fact that the input device is usually a video device, which can be easily adjusted and focused on the center of the frame.

Learning the sea pattern can be done by an occurrence matrix. The occurrence matrix is $N \times N$, where N is the possible number of gray levels inside the frame [9].

We define a length function f from current pixel (x, y):

$$f(x) = x + l_d \tag{8}$$

where l_d is the distance to the proportional pixel. Eq. 8 is used to determine the occurrence matrix values, depending only on x values based on the sea texture assuming that the sea texture is horizontal. In the next stage we scan the image and set the pixel values the following way:

1. For each pixel in the learning zone, check if this pixel is inside the search space
2. If so, set pixel (x,y) value to k
3. Choose the next pixel according to f value

The occurrence matrix allows us to find similar texture inside the image. The learning zone can vary from image to image as shown in Figure 3. For an (n,m) image we use 10 percent of n on each side (left and right), and 10 percent of m at the bottom as learning zone.

Fig. 3 Learning Sea Pattern: on the left we can see the original image and the right side shows the image after applying occurrence matrix

3.4 Morphologic Cleaning

After completing the detection process of the sea pattern, there are still few sea pixels that were not detected. We use a unique cleaning filters to improve and smoothen the image, so target identification will be easier. The first filter is the "Fill Filter". This filter fills the target with missing pixels, i.e. for each black pixel (sea pixel). We scan the eight neighbors of each pixel. If the pixel's eight neighbors are white (target pixels), we change the pixel value to be a target pixel.

We use the Fill Filter Matrix, F_f:

$$F_f[i,j] = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{9}$$

The second filter is called "Cleaning Filter". This filter makes exactly the opposite action from the Fill Filter. For each white pixel (target pixel), if all of its eight neighbors are black (sea pixels), we change this pixel value to be a sea pixel.

The Cleaning Filter Matrix C_f is:

$$C_f[i,j] = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{10}$$

The third filter is called "Connection Filter". In some cases Fill filter and Cleaning filter can not recover damaged pixels if the problem occurres in more than one adjacent pixel. Therefore, the Connection filter detects two separate components of a target and connects them. The Connection Filter Matrix C_{nf} is:

$$C_{nf}[i,j] = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \tag{11}$$

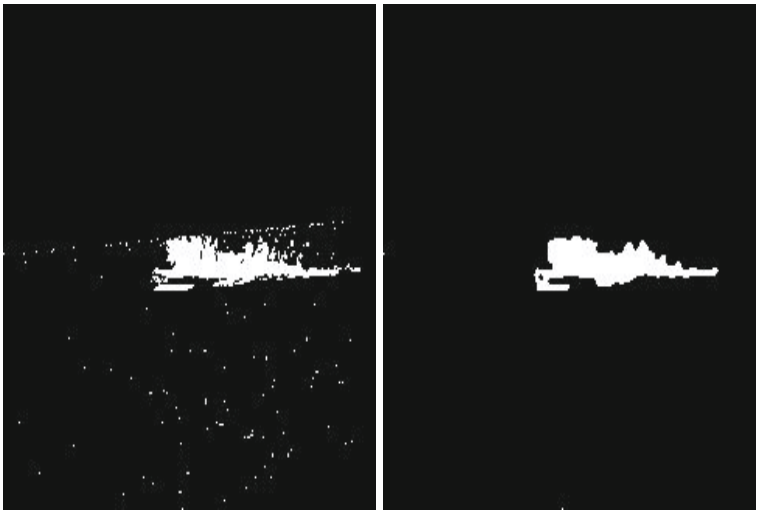


Fig. 4 Morphologic Cleaning: The left side shows the image after detecting the sea pattern and the right side shows the image after applying the morphologic methods: Cleaning and Fill filter

The forth filter is based on the assumption that each pixel value is determined by its neighbor pixel values. This filter is called "Majority Filter". For each pixel, is calculates the eight neighbor values and changes the current pixel value according to the majority.

The left side of Figure 4 shows the image after detecting the sea pattern and before applying filters. The right side of Figure 4 shows the image after applying the cleaning filters. We can see that all the damaged pixels were removed and we managed to derive a clean target image.

3.5 Marking Interesting Points

The video device and the detection algorithm are not perfectly accurate and damaged pixels will always be a part of the image. We assume that the basic structure of the target does not change from one frame to another. The algorithm characterizes a target by using a Skeleton algorithm. This algorithm applies a thinning process to an image and finds a simpler structure of the recognized targets. This process simplifies the target tracking in the next frames and decreases the CPU time.

We use an extended version of the Skeleton algorithm. The extended version is based on doubled thinning process of the target. For each pixel x belongs to the thin target profile, the pixel at $x+1$ location also marked as a pixel in a thin profile of the target.

The left side of Figure 5 shows the target image before the thinning process and the right side shows the thin target after applying the Skeleton algorithm.

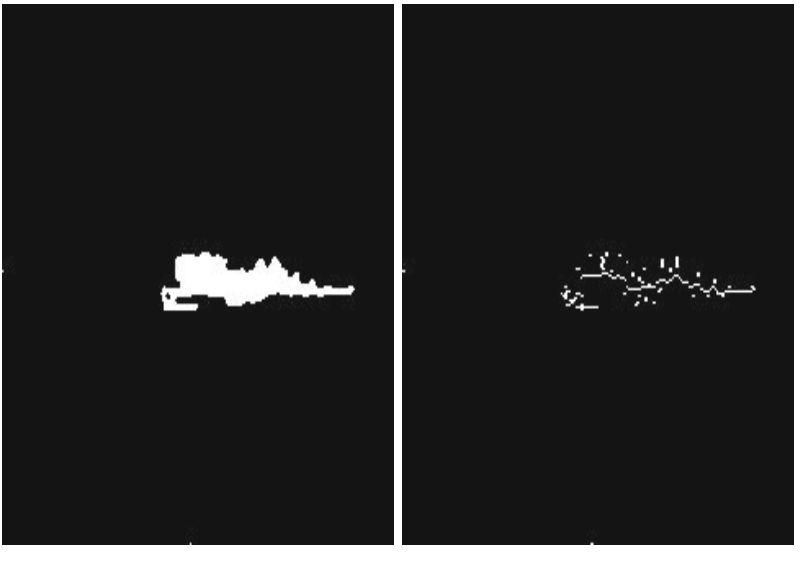


Fig. 5 Skeleton Algorithm: The left side shows the target image before the thinning process and the right side shows the thin target after Skeleton algorithm without Interesting Point

Marine environment can be unstable and thin structure targets usually can't be seen in the frame. Therefore, the target structure should be more flexible, supporting target recognition even if not all of the target is in the frame. For that, we define the Interesting Point concept:

Definition: Interesting Point - is a point in the target thin profile which is an intersection of two other lines. i.e. considering thin target, for each white pixel (Skeleton pixel) we perform the Convolution Operator C_o with the matrix:

$$C_o[i,j] = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{12}$$

The matrix C_o identifies most significant points within a two line intersection with a predefined threshold S_t in the Skeleton algorithm.

As mentioned before, CPU time is a major issue in our algorithm implementation. Tracking targets in the next frames is based on a simple structure of the target, reducing CPU time. Moreover, it is much more accurate to characterize a target with a few points, rather than using a large number of points in such a clutter and noisy environment.

Figure 6 shows the skeleton image after the thinning process, the left side demonstrate the target interesting points implementation.

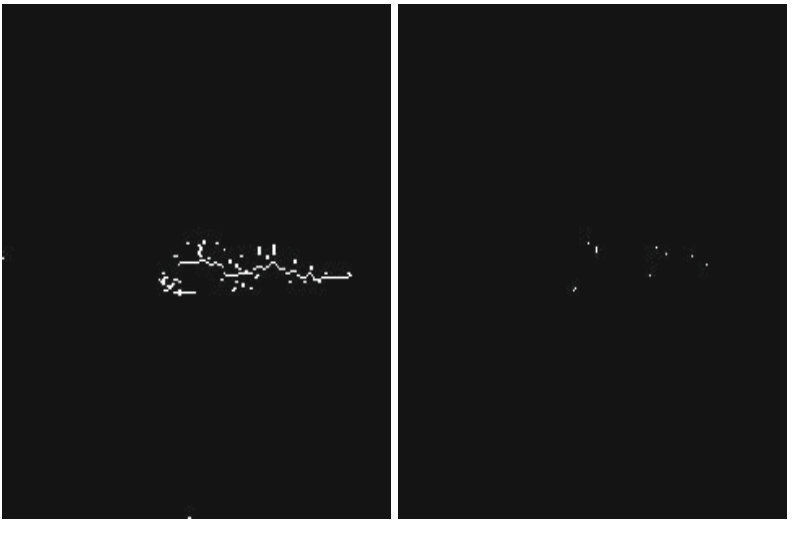


Fig. 6 Skeleton Algorithm with Interesting Points: The left side shows the target image after Skeletone thinning process and the right side shows the Skeleton algorithm with Interesting Point

3.6 Time Correlation

This is the final step in the recognition algorithm part. After we have achieved two sequential frames and identified the targets in each one of them, we apply the time correlation step. Our goal is to make the final decision about a possible target that was found in the first frame and to decide whether it is a real target or a false alarm.

Given a set of interesting points in the first frame $N = n_1, , n_n$, and another set of interesting points in the second frame $M = m_1, , m_m$. For each interesting point $n_1, , n_n$, we build a surrounding rectangle of size *AxA*. In the second frame, we locate a *BxB* rectangle at the same place around the interesting point. In this stage, we start to search the interesting point of the smaller rectangle inside the bigger rectangle. If the interesting point is found, it will appears in the sequential frame. The same search process is implemented for all of the interesting points $n_1, , n_n$ in the same way. The target position is determined by the average of all it's interesting points.

Figure 7 shows the first frame interesting points. The red rectangle is marked and all of the interesting points inside it are registered. The next step is to build the blue rectangle in the sequential frame and search all of the registered interesting points inside. Once 80 percent of the matching is complete, we declare those points as relevant and continue searching the next points as a candidate target.

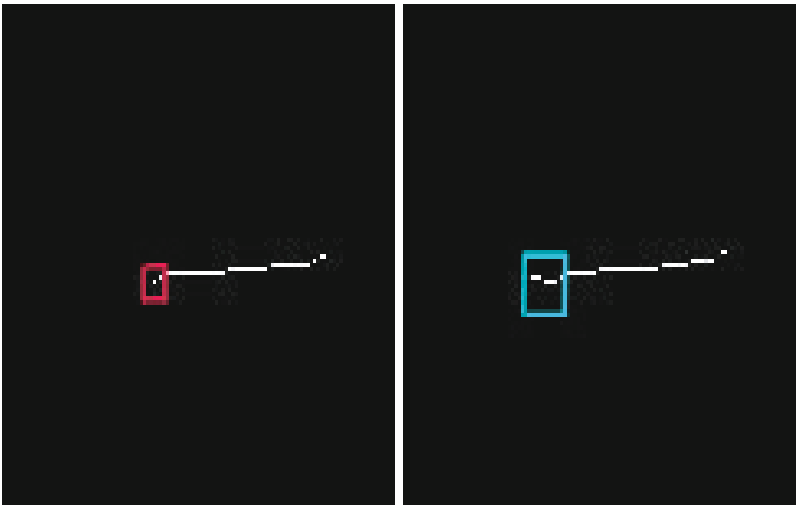


Fig. 7 One Step of Time Correlation Process: In the left side, the interesting points are marked with red rectangle and in the right side the next time frame introduced with blue rectangle searching the same interesting points from the previous frame

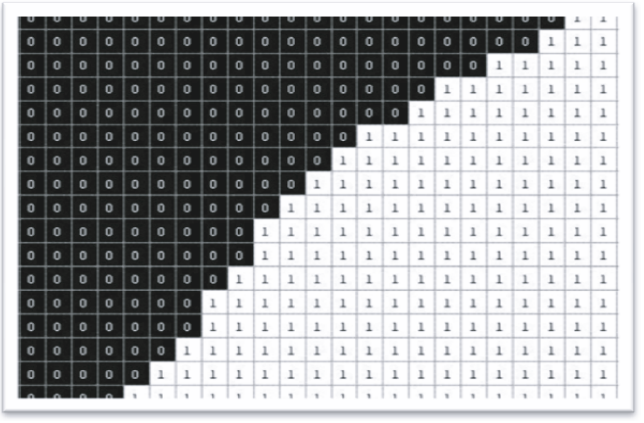


Fig. 8 Labeling Algorithm Presentation: Each label is colored with a different color, in this case there is only one target colored in red

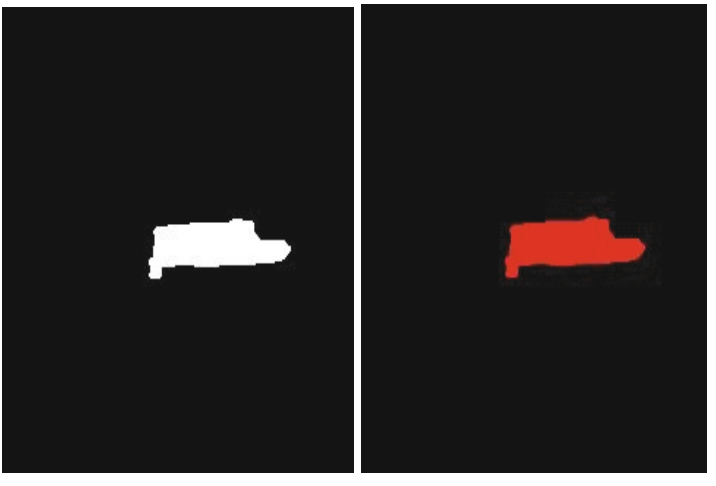


Fig. 9 Labeling Algorithm Matrix: $n \times n$ matrix for each target, each cell in the matrix $[i, j]$ defines the connected component related to the interesting point, in this case we classify only one target with serial numbers of 1 and 0

3.7 Multiple Targets

One of the main goals of the algorithm is the ability to track multiple targets at the same time. This can be done by classifying the interesting points. Classification of the interesting points is based on connected components using labeling algorithm. The input image for the labeling algorithm is a target image after morphologic cleaning. In case of multiple targets, the labeling algorithm recognize and label each

component in a different serializing number. Then, each interesting point is classified according to its matching number. Example of labeling algorithm matrix $n \times n$ can be seen in Figure 8. Each cell in the matrix $[i, j]$ defines the connected component related to the interesting point. This method enables us to track multiple targets in an efficient way.

Figure 9 shows the output of the labeling algorithm. Each label is colored with a different color, in this case a single component colored in red.

4 Results

Our algorithm was tested with five hours of video stream from a database with real video recorded with a camera on a USV in different light and sea states. The algorithm was tested on real-time video movies from the USV system, the first frame can be seen in Figure 10.

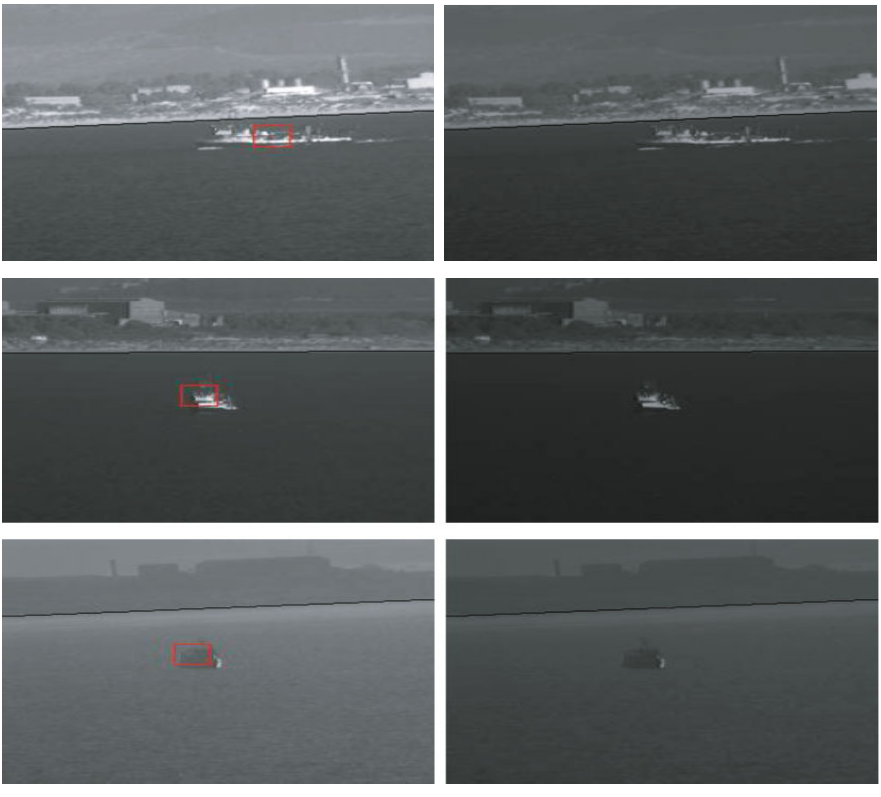


Fig. 10 Real-time Video Algorithm Performance: Real video camera on USV in different light and sea states and targets radius. Selected movies can be found on-line at [13].

Target recognition and identification marked with a red rectangular and can be found at: *http://sites.google.com/site/orenusv/home/research-1/usvvision.*

The parameter values in our simulation are: $l_d = 10$, the lower threshold value was set to 20 and the upper value was set to 100, the surrounding rectangles size were set to $A = 7, B = 16$ and $fac = 3$, $S_t = 4$.

The tested scenes contained different sea states at day and night with different light conditions. The target radius changes between $10 - 60[m]$. The horizon line was examined with and without buildings in the background of the USV.

The run-time for average on frame was measured as 0.005 millisecond, with the minimum value 0.003 millisecond and the maximum value 0.014 millisecond. Based on these results, the algorithm can be used for real-time application for autonomous movement of USV, identifying targets around the vehicle.

5 Discussion and Conclusion

This paper presents a basic and very efficient algorithm for marine environments that was tested on different scenes. Yet, there are some limitations and untested cases that should be treated in future research. Our algorithm is limited to sensor noises and false alarm dealing with very cluttered environments. The algorithm has been tested with stabilize sensor on the platform, which simplify horizon line recognition. In case of not stabilized sensor or panoramic image integrating several video streams, we expect a very limited success rate.

Another well know problem in marine environments is related to sun effects. The algorithm might classify sun light effects as targets, which will cause false alarms. This challenge should be treated in our further research, for more accurate ATD abilities for accurate USV trajectory planning.

The highlight of the algorithm is the simple concept that can be use on COTS sensor without special hardware, and as far as we know this specific issue was not yet studied extensively in marine environments. The algorithm is based on simple basic algorithms from the image processing world, which are suitable for real-time application.

We presented a new algorithm to acquire identify and track obstacles from USV systems using COTS sensor for autonomic navigation. The algorithm is based on previous image processing filters and algorithms, and been been adapted to the marine environment challenges.

The algorithm was successfully tested on real-time video from USV systems, and can be apply in real-time applications dealing with CPU time constraints. Further research directions include light effects and not stabilized systems.

Acknowledgements. The author would like to thank Yaron Vazana for his help during this research and to Dr. Blaurock and Dr. Schlaefer for their helpful comments.

References

1. Bhanu, C., Holben, R.D.: Model-based segmentation of FLIR images. IEEE Trans. Aerosp. Electron. Syst. 26, 2–10 (1990)
2. Ben-Yosef, N., Bahat, B., FeiginBhanu, G., Holben, C.: Simulation of IR images of natural backgrounds. Appl. Opt. 22, 190–193 (1983)
3. Ratches, J.A., Walters, C.P., Buser, R.G., Guenther, B.D.: Aided and automatic target recognition based upon sensor inputs from image forming systems. IEEE Trans. Pattern Anal. Mach. Intell. 19, 1004–1019 (1997)
4. Casasent, D.P., Neiberg, L.M.: Classifier and shift-invariant automatic target recognition neural networks. Neural Networks 8, 1117–1129 (1995)
5. Schachter, B.J., Lev, A., Zucker, S.W., Rosenfeld, A.: An application of relaxation methods to edge reinforcement. IEEE Trans. Syst. Man Cybern. 7, 813–816 (1997)
6. Walters, D.K.W.: Computer vision model based on psychophysical experiments. Pattern Recognition by Humans and Machines 2 (1986)
7. Broy, M.: Early vision. In: Rosenfeld, A. (ed.) Perspectives in Computing, pp. 190–206. Academic Press, New York (1986)
8. Marham, K.C.: Comparison of segmentation processes for object acquisition in infrared scenes. IEEE Radar Signal Process 136, 13–21 (1989)
9. Davies, E.R.: Machine Vision: Theory, Algorithms, Practicalities, 2nd edn. Academic Press, New York (1997)
10. Marham, K.C.: Clutter metrics for target detection systems. IEEE Trans. Aerosp. Electron. Syst. 30, 81–91 (1994)
11. Vijaya, K.: A tutorial survey of composite filter designs for optical correlators. Appl. Opt. 31, 4773–4798 (1992)
12. Flannery, D., Loomis, J., Milkovich, M.: Transform-ratio ternary phase-amplitude filter formation for improved correlation discrimination. Appl. Opt. 27, 4079–4083 (1988)
13. http://sites.google.com/site/orenusv/home/research-1/usvvision

Rule-Compliant Navigation with Qualitative Spatial Reasoning

Diedrich Wolter, Frank Dylla, and Arne Kreutzmann

Abstract. We develop a formal, symbolic representation of right-of-way-rules for sea navigation based on a qualitative spatial representation. Navigation rules specified qualitatively allow an autonomous agent consistently to combine all rules applicable in a context. The focus of this paper is to show how the abstract rule specification can be used during path-planning. We propose a randomized-qualitative approach to navigation, integrating the symbolic level with a probabilistic roadmap planner. The resulting navigation system maneuvers under the side constraint of rule compliancy. Evaluating our approach with case studies we demonstrate that qualitative navigation rules contributes to autonomous sailing.

1 Introduction

A considerable amount of everyday behavior is not self-determined but subject to regulations. For example, right-of-way regulations govern how to travel public spaces. Action planning for an autonomous agent needs to respect right-of-way regulations. These rules are special in that they have been designed for the general public and are denoted in natural language, using abstract concepts of space. Making these regulations accessible to an artificial agent requires translating them into a formal language that can be understood by the agent and which seamlessly integrates with the agent's navigation process. In order to facilitate correctness and verifiability of the translation, an abstract language is particularly suited if it is able to reflect the concepts originally used in the right-of-way regulations. We use *qualitative representations* to abstract real-wold observations to abstract knowledge about

Diedrich Wolter · Frank Dylla · Arne Kreutzmann
SFB/TR 8 Spatial Cognition, University of Bremen
e-mail: {dwolter,dylla,kreutzma}@sbftr8.uni-bremen.de

space and time on a conceptual level. Qualitative spatial representations (see [4] for an overview) aim to provide a formal model for human-level commonsense understanding of space and time. Moreover, they enable abstract reasoning processes. Technically, qualitative representations summarize similar real-world states by a discrete, finite set of qualitative categories that give rise to symbolic reasoning.

This paper demonstrates the utility of qualitative reasoning in autonomous sea navigation. In previous work we have studied how purely symbolic reasoning can help to consistently integrate pair-wise rule constraints when multiple agents meet [6]. We now focus on the problem of actually controlling a vessel in a rule-compliant manner. The contribution of our work is to show how the official right-of-way rules for vessel navigation (COLREGS: vessels in sight of each other) according to the International Maritime Organization (IMO) can be modeled using qualitative spatial representation. Furthermore, we show how the representation supports rule-compliant action planning for autonomous vessels. This paper is organized as follows. We start by putting our approach in the context of high-level agent control. Section 3 introduces qualitative representation and reasoning techniques. Using these techniques, Section 4 details our formalization of navigation rules. Section 5 explains how we incorporate the qualitative rules into action planning. We give an experimental account of our approach in Section 6 and conclude the paper by summarizing the results and discussing further research directions.

2 Rule-Compliant Navigation

Rule-compliant navigation starts by formalizing navigation rules in a formal language that can be understood by autonomous agents. To this end, symbolic navigation rules are suitable [11], in particular qualitative representations can be incorporated in logic-based agent control [3]. Such techniques tackle planning only on the level of abstract actions though. Navigation rules, in particular codes of practice for sailing, can also be captured in Fuzzy representations [16], but they lack the formal semantics that allow abstract processes to reason about the consistency of actions possible with qualitative representations [6]. Navigating by qualitative rules requires bridging from abstract spatial relations to concrete control parameters needed by the actuators of the robotic system. Thus, symbolic reasoning needs to be linked to control parameter selection. The example of tacking, a complex turn maneuver in sailing (see Fig. 1), illustrates the difficulty of this integration. Tacking requires several preconditions to be met in order to perform the maneuver, for example, enough free maneuver space needs to be available. The amount of space required depends on the specific physical context like wind, initial vessel speed, inertia of the vessel, etc. If the initial speed of the vessel is too slow, tacking fails. It appears to be difficult to come up with an abstract definition of tacking that is precise enough to represent exactly those situations in which the maneuver is possible. For example, overestimating the space requirements may inhibit planning to identify situations in which the action can be performed, underestimating it may lead to accidents. Thus, the applicability of symbolic planning can be questioned.

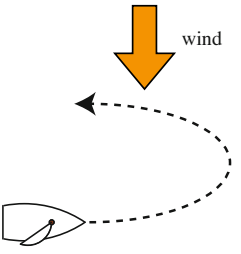


Fig. 1 Tacking a sailboat

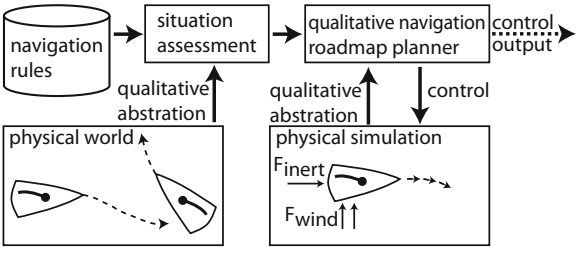


Fig. 2 Architecture overview

By contrast, we use the symbolic level to formalize navigation rules as sequences of key configurations to pass through, avoiding action definitions. Key configurations are used as intermediate goals in a probabilistic roadmap planner. Probabilistic planning has previously been shown to be applicable in traffic planning. In contrast to the approach by [14], we explicitly model collision regulations in a formal language. The combination of qualitative representation with probabilistic roadmap planning has also been suggested in [17], but our approach does not need to generate a plan on an abstract level before invoking the action planner. This way we avoid the aforementioned problem of exactly describing action preconditions. Figure 2 presents an overview of our approach. Based on the qualitative assessment of an observation we select the navigation rules applicable to the situation. Then we employ the planner to determine control actions for rudder and sheet rope length that allow the vessel to navigate under the side constraints of rule consistency. Our approach can be regarded as a hybrid navigation system involving mathematical models in the terminology of Statheros et al. [15], which argue for the use of hybrid models in ship collision avoidance.

3 Qualitative Spatial Knowledge Representation

Qualitative Spatial Reasoning (QSR)[1] is the subfield of knowledge representation involved with spatial representations that abstract from the details of the physical world. Its reasoning techniques allow predictions about spatial relations, even when precise quantitative information is not available [4]. Based on qualitative representation and corresponding reasoning methods computers can be enabled to monitor, diagnose, predict, plan, or explain the behavior of physical systems [9]. In general, two categories of reasoning based on qualitative spatial representations can be distinguished: *constraint-based reasoning* to reason about static configurations and *neighborhood-based reasoning* to reason about how qualitative representations can

[1] As reasoning is not possible without representation we will not distinguish between them generally in the remainder of this paper. That is, we shall refer to qualitative spatial representation or qualitative spatial reasoning only.

change over time. In the following we give an intuitive introduction to the basic concepts of QSR—the interested reader is pointed to the literature [4, 13].

A central notion in QSR is that of a *qualitative calculus* which comprises a finite set of *atomic relations* to describe the relationships between entities as well as operations on these relations. Technically, these relations are binary (sometimes ternary) relations between domain level objects. In this paper we only consider binary relations. Relations feature a set-theoretic semantics, i.e., a binary relation r on the domain D is a subset $r \subseteq D \times D$, i.e., a set of ordered pairs (x, y) with $x, y \in D$. The set R of all relations of a qualitative calculus is assumed to be jointly exhaustive and pairwise disjoint. Hence, we have a boolean set algebra with the usual set operations. Elements of this set algebra, i.e., arbitrary disjunctions of atomic relations, are also called *qualitative relations* for short.

Qualitative calculi define two relation operations which allow new facts to be derived from given ones: *conversion* and *composition*. Mathematically, qualitative calculi relate to relation algebras in the sense of Tarski. *Conversion* can be interpreted as shifting perspective from one entity to another. For example, conversion allows us to infer from the fact that Lübeck is *NortEast* of Bremen that it also holds that Bremen is *SouthWest* of Lübeck. The *composition* operation allows knowledge about a common entity to be combined. Only using that Hamburg is *NortEast* of Bremen and Lübeck is *NortEast* of Hamburg, we can still derive with composition that Lübeck is *NorthEast* of Bremen. Since QSR is involved with finite sets of atomic relations only, the composition operation is usually provided in form of look-up tables called *composition tables*. These operations are particularly important for constraint-based qualitative reasoning [13] and often enable efficient algorithms [12] to tackle the problem of deciding whether a set of constraints involving qualitative relations is consistent or not.

Conceptual neighborhood extends static qualitative representations by interrelating the discrete set of base relations [7]: *Two spatial relations of a qualitative spatial calculus are conceptually neighbored, if they can be continuously transformed into each other without resulting in a third relation in between.* We note, conceptual neighborhood on the qualitative level corresponds to continuity on the physical level. For example, let us consider the relations *behind*, *same*, and *ahead* to relate the positions of two vessels in a match race. For reasons of simplicity we assume that vessels are only able to move forward with changing speed. In the leftmost configuration shown in Fig. 3 vessel **A** is *behind* **B**. Observing the scene a few minutes later shows that now **A** is *ahead* of **B**. Assuming continuous motion it is not possible for **A** to overtake **B** without passing **B** at some time, i.e. being at the *same* level. Therefore, *ahead* and *behind* are not conceptually neighbored, whereas *ahead* and *behind* are both conceptual neighbors of *same*.

In order to apply conceptual neighborhoods for reasoning about actions, it is helpful to label neighborhood transitions with actions that initiate the respective transitions. The resulting structure is called the *action-augmented conceptual neighborhood* [5].

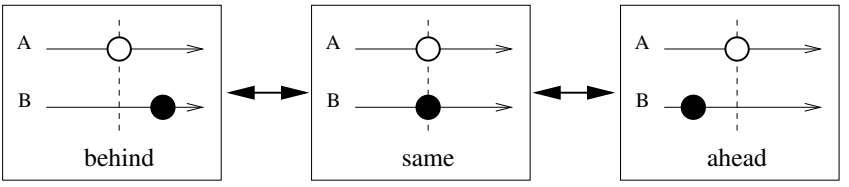


Fig. 3 Ordering relations arranged their conceptual neighborhood relationship

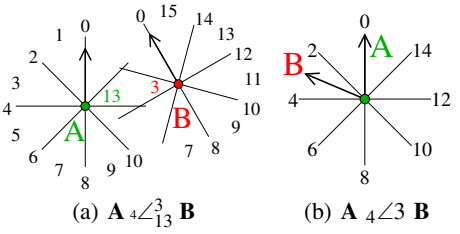
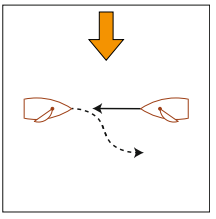


Fig. 4 Two oriented points related at granularity $m = 4$

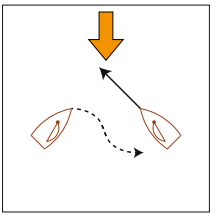
3.1 A Qualitative Calculus of Relative Agent Position

Navigation rules are often formulated in an egocentric frame of reference. For example, the notion "oncoming traffic" refers to traffic traveling in the direction opposite to that of the observer. In order to represent such knowledge we require a qualitative calculus about directional information. We base our formalization on the $\mathcal{OPRA}_m$ [10] calculus which describes relations between object in the domain of oriented points in the plane, i.e., 2D points equipped with a direction. $\mathcal{OPRA}_m$ relations describe the position of an object B as seen from A and, simultaneously, the position of A as seen from B. Relations are thus pairs of directions, written $\mathbf{A}\,{}_m\angle^j_i\,\mathbf{B}$ where i, j denote the directions. The calculus is defined for an arbitrary granularity $m \in \mathbb{N}$, controlling how many directions are to be distinguished. In our work we use $m = 4$ (Fig. 4). For each of the two related oriented points, m lines are used to partition the plane into $2m$ planar and $2m$ linear regions. Direction 0 is aligned with the orientation of the point. If two positions coincide, so-called *'same' relations* occur. In these cases a number s denotes the direction in which B is oriented, as seen from **A** (Fig. 4(b)), written as $\mathbf{A}\,{}_m\angle s\,\mathbf{B}$.

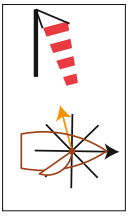
$\mathcal{OPRA}_4$ provides no atomic front or back region, as it is needed to represent 'head-on', for example. Such region can easily we described as disjunctions of finer-grained $\mathcal{OPRA}_8$ relations. To ease notation we define $\mathcal{OPRA}^{\mathrm{x}}_4$ relations as a shorthand notation for such disjunctions. Geometrically, $\mathcal{OPRA}^{\mathrm{x}}_4$ relations can be obtained as follows: we rotate the segment boarders of $\mathcal{OPRA}_4$ by half of the angular resolution, i.e., by $\frac{1}{2} \cdot \frac{360^\circ}{2m} = 22.5^\circ$, and join the linear regions with the planar ones. In Fig. 6 we depict an example of relation $_4\mathrm{x}^0_0$.



(a) head-on: $\mathbf{A}\ {}_4x_0^0\ \mathbf{B}$



(b) crossing: $\mathbf{A}\ {}_4x_7^1\ \mathbf{B}$



(c) port-side wind: $\mathbf{A}\ {}_4\angle 5\ \mathbf{A}_{\text{wind}}$

Fig. 5 Two sailing vessels in a head-on and a crossing situation and the advised collision avoidance behavior

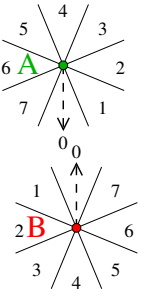


Fig. 6 An example relation for $\mathcal{OPRA}_4^x$: $\mathbf{A}\ {}_4x_0^0\ \mathbf{B}$

4 Formalizing Navigation Rules

The basis for our rule formalization is formed by parts of the *International Regulations for Avoiding Collisions at Sea* (COLREGS, simplified rules 11–18). These rules describe how vessels have to behave if they are in sight of each other in order to avoid collisions. The rules in the COLREGS are given in natural language using abstract spatial terms. Additionally, textbooks show pictorial representations in order to give people a more vivid interpretation about conditions and execution of rules. For deriving rule formalizations we follow the approach taken by Dylla et al. [5, 6], which employs $\mathcal{OPRA}_m$ and its conceptual neighborhood structure. In extension, we give here additional qualitative representations and consider sailing vessels as well.

Let us consider rule 12 (a) and its pictorial representations (Fig. 5) in head-on and crossing situation:

> Rule 12:
> (a) when two sailing vessels are approaching one another, so as to involve risk of collision, one of them shall keep out of the way of the other as follows:
>
> (i) when each of them has the wind on a different side, the vessel which has the wind on the port side shall keep out of the way of the other;
>
> (ii)...

In order to define *rule-consistent* or *rule-compliant* behavior we need to operationalize the individual rules. First, we translate and ground the natural language terms in qualitative relations and, second, formalize the rules by means of this representation. Finally, we will define rule-compliant behavior based on these formalizations. By using $\mathcal{OPRA}_m$ relations we abstract from the physically extended objects to oriented points. Wether the extended objects collide or not will be handled in the planning phase.

The start configurations for the application of rule 12 (a) is that 'two sailing vessels are approaching one another, so as to involve risk of collision'. For example, this configuration is given if two vessels are meeting on reciprocal or nearly reciprocal courses, which is described by relation ${}_4\mathrm{x}_0^0$.[2]

Furthermore, we need to know from which side of the vessel the wind comes from. We represent the orientation of the wind by an $\mathcal{OPRA}_4$ 'same' relation which is generated from the vessel's heading and the orientation of the wind as seen from the center point of the vessel. In contrast to the representation of the relative orientation of the vessels we need the exact heading. Therefore, we cannot apply $\mathcal{OPRA}_4^{\mathrm{x}}$ and must apply the original $\mathcal{OPRA}_4$. Thus, if $\mathbf{A}\ {}_4\angle i\ \mathbf{A}_{wind}$ with $i \in \{1,..,7\}$ than the wind is coming from port and with $i \in \{9,..,15\}$ its coming from starboard. In summary, the conditions of rule 12 (a) can be represented as

$$\mathbf{A}\ {}_4\mathrm{x}_0^0\ \mathbf{B} \wedge \mathbf{A}\ {}_4\angle i\ \mathbf{A}_{wind} \wedge \mathbf{B}\ {}_4\angle j\ \mathbf{B}_{wind} \tag{1}$$

with $i \in \{1,...,7\}$ and $j \in \{9,...,15\}$. The evasion behavior needs to be modeled next. Vessel **A** is the give-way vessel and **B** the stand-on vessel. Regarding the advised behavior as shown in Fig. 5(a), **B** must keep its course and **A** must turn starboard in order to avoid a collision. Considering action-augmented conceptual neighborhood this results in a changeover to relation ${}_4\mathrm{x}_1^0$. After the turn it is a reasonable strategy for **A** to move just about straight on, which leads to relation ${}_4\mathrm{x}_1^1$. Going on like this brings us to relation ${}_4\mathrm{x}_2^2$ and ${}_4\mathrm{x}_3^3$ subsequently. At this point the rule could be considered to be successfully performed, but to acknowledge that **A** should return to its original course we also add relation ${}_4\mathrm{x}_4^4$. In summary, we represent rule 12 (a) as the sequence (denoted $\rightarrow$) of formulae (see also Fig. 7):

$$\begin{aligned} &\mathbf{A}\ {}_4\mathrm{x}_0^0\ \mathbf{B} \wedge \mathbf{A}\ {}_4\angle i\ \mathbf{A}_{\mathrm{wind}} \wedge \mathbf{B}\ {}_4\angle j\ \mathbf{B}_{\mathrm{wind}} \\ &\qquad\qquad \text{with } i \in \{1,...,7\} \text{ and } j \in \{9,...,15\} \\ &\rightarrow \mathbf{A}\ {}_4\mathrm{x}_1^0\ \mathbf{B} \rightarrow \mathbf{A}\ {}_4\mathrm{x}_1^1\ \mathbf{B} \rightarrow \mathbf{A}\ {}_4\mathrm{x}_2^2\ \mathbf{B} \rightarrow \mathbf{A}\ {}_4\mathrm{x}_3^3\ \mathbf{B} \rightarrow \mathbf{A}\ {}_4\mathrm{x}_4^4\ \mathbf{B} \end{aligned} \tag{2}$$

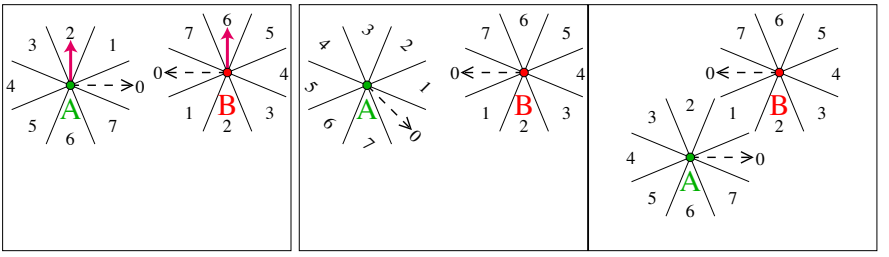
Finally, based on such rule descriptions we define *rule-compliant* behavior. We assume rule $R^x = r_0^x \rightarrow ... \rightarrow r_{n_x}^x$ as given with x being the rule number.

Definition 1. *Admissible configuration:* In context of a rule $R^x = r_0^x \rightarrow \ldots \rightarrow r_{n_x}^x$ only configurations r_i^x included in the rule are admissible.

Definition 2. Agent behavior is *rule-compliant* or *admissible* wrt. R^x if:

1. the vessels' initial configuration is r_0^x
2. during rule execution the vessels are only in admissible configurations
3. only changes from r_i^x to $r_{i\pm1}^x$ occur and r_{i-2}^x does not occur after r_i^x

[2] We are aware that this representation also includes situations where agents are far apart and no risk of collision is given. For reasons of simplicity we do not to exclude these cases by a refined representation here.



(a) (b) $\mathbf{A}\ {}_4\mathrm{x}_1^1\ \mathbf{B}$ (c) $\mathbf{A}\ {}_4\mathrm{x}_3^3\ \mathbf{B}$

head-on: $\mathbf{A}\ {}_4\mathrm{x}_4^0 0\ \ \mathbf{B}$
$\wedge\ \mathbf{A}\ {}_4\angle 4\ \ \mathbf{A}_{\text{wind}}$
$\wedge\ \mathbf{B}\ {}_4\angle 12\ \ \mathbf{B}_{\text{wind}}$

Fig. 7 Iconographic representation of the first steps in formalizing Rule 12 (a)

4. the vessels are in configuration $r_{n_x}^x$ at the end
5. no collision occurs

Other formalizations, for example, following shipping routes defined by buoys and light signals can also be formalized using the same approach. Currently, we do not regard disjunctive rules as, for example in overtake situations where one is allowed to pass port or starboard. The extension to include such formalizations is straightforward though.

5 Navigation by Qualitative Rules

In our approach qualitative representations serve exclusively for categorizing a configuration and for checking rule-consistency. A randomized planner generates hypotheses of actions to perform, only requiring the forward kinematics of the agent (given by a sailing simulator in our case). Actions generated by the planner are then assessed qualitatively—actions that violate rules are discarded and the most promising actions are further considered. Although a planner may be capable of determining complex action sequences, it is advisable only to exectute the first actions and to re-plan as soon as possible. Continuous re-planning allows the system to respond to unforeseen situations, for example changing winds or unexpected behavior of others.

5.1 Probabilistic Roadmap Planner

In our approach action selection is performed by a *probabilistic* or *randomized roadmap planner (PRM)* [8]. This type of planners is particularly helpful for motion planning when no inverse kinematic model is given, only a forward kinematic or simulation is needed. Another feature of interest is its ability to incorporate further constraints, such as scoring solutions by the intermediate locations visited or

efficiently re-computing paths in dynamic environments [2, 1]. In a nutshell, a PRM builds a graph of the search space similar to classical AI search techniques. Nodes in the graph represent states of the search space, they are linked by edges that are labeled by the action that allow an agent to get from one state to the next. The objective is to determine a path from the start node to a goal node. During planning, a node is randomly selected and expanded by performing a fixed number of random expansions, i.e., random actions are performed. A heuristic scoring function h is employed to rate the expansion probability of a node and to facilitate goal-directedness.

We represent the dynamic state of the vessel as nodes which are then linked by the rudder and sail actions performed. Also, we record the complete trajectory from the start position to the respective node as well as the total plan duration measured in simulation time. Every node representing a vessel position that is closer than 10m to the desired goal position is considered to be a goal state. We use bold to denote vectors and $\langle\cdot,\cdot\rangle$ for the scalar product. Our heuristic scoring function contoling random node selection is based on the position $\mathbf{p}$ of a vessel, its velocity vector $\mathbf{v}$, and the goal position $\mathbf{g}$.

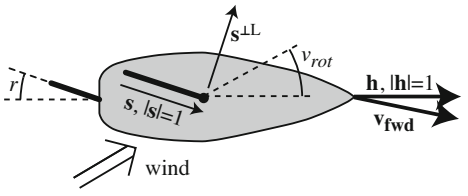
$$h(n) := \begin{cases} 0 & \text{, the trajectory of } n \text{ is not rule-compliant} \\ h' & \text{, otherwise} \end{cases} \tag{3}$$

$$h' = \langle \mathbf{p}-\mathbf{g}, \mathbf{p}-\mathbf{g} \rangle \cdot (1 + \max\{0, \langle \mathbf{v}, \mathbf{p}-\mathbf{g} \rangle\}))^2 \tag{4}$$

Any node corresponding to a trajectory that is not rule-compliant is assigned a score of zero, i.e., it cannot be selected any more for expansion. This ensures that the planner always determines a rule-compliant plan. For rule-compliant nodes, the scoring combines distance to the goal with a speed component (second term in Equation 4). This term serves to differentiate positions that are similarly close to the goal, but in which the vessel is either sailing towards the goal or away from it.

Random node selection first determines the total score $s = \sum_{i=1}^{o} h(N_i)$ of all open nodes $N_1, N_2, \ldots, N_o$ and then samples a uniformly distributed random number r in the interval $[0, s]$, selecting the node N_j with the smallest value of j such that $\sum_{i=1}^{j} h(N_i) > r$. If a node is selected, n random actions are generated and the search graph is expanded.

In order to avoid combinatorial explosion that would occur if continuously expanding nodes, we restrict the size of the set of active nodes which can be further expanded. After a series of node expansions, the set of active nodes is sampled to cut it down to its initial size. Doing so, the memory requirement of the planner is kept constant. We perform the sampling simply by first selecting all k nodes to expand as explained above and then performing node expansion. Nodes that have not been selected are discarded immediately. Although limiting the set of active nodes may discard states that lead to the goal, the step is necessary to obtain a method that can generate a plan using limited computational resources. In our evaluation we analyze different choices for the size of the set of active nodes in order to identify a good balance between the ability of the planner to determine a path and memory requirements.



symbol	value	description
c_{sail}	50	size of sail in m^2
c_{m}	10000	mass in kg
c_{hull}	0.02	coefficient longitudinal vs. lateral friction of hull
c_{up}	0.5	buoyant force
c_{drift}	0.001	wind drift
c_{fr}	0.005	friction
c_{rot}	100	rotational inertia
c_{r}	5.0	rudder force coefficient

Fig. 8 Legend of variables and parameters used in the physical simulation

5.2 Physical Simulation

Designing our simulation we aimed to create a mock-up of the sailing experience with cruising yachts (approximately 10m of length, deep single-fin keel, and a single mainsail). As control commands we only consider the position of the rudder and the length of the sheet rope that controls how far the boom is opened. We employ a very efficient but idealized physical simulation to determine the effects of actions and development of the environment. Most essentially, we make the following idealization:

- No waves, no fluid simulation
- Simple wind model, no turbulence, no slip streams
- Control actions are performed in a single simulation step

Although true sailing sport draws its attraction from some of these facets, we believe that they can be neglected in context of navigating in safe operation range. Unfortunately, any realistic simulation involves careful modeling of physical phenomena beyond the scope of our work and it would require considerable computational resources. Since roadmap planners make intense use of the simulation, its efficient implementation is key. We employ the variables shown in Fig. 8 to describe the dynamic physical state of any vessel.

By $\mathbf{v}^{\perp}$ we refer to the left normal vector of a vector $\mathbf{v}$ and $\mathbf{v}^{\perp \text{L}}$ stands for the lee site normal vector. In order to model the water resistance we decompose friction into friction along its longitudinal and its lateral axis.

$$d(\mathbf{f},\mathbf{h},\alpha,\beta) := \alpha\langle \mathbf{f},\mathbf{h}\rangle \cdot \mathbf{h} + \beta\langle \mathbf{f},\mathbf{h}^{\perp}\rangle \cdot \mathbf{h}^{\perp} \tag{5}$$

Based on the current local wind vector $\mathbf{w}$ obtained as the difference between global wind and current speed we determine the resulting acceleration, which is then integrated in a constant time-step simulation to update the dynamic vessel parameters.

$$\mathbf{a_{fwd}} := \frac{1}{c_m}\Big(\underbrace{d(|\mathbf{v_{fwd}}|\cdot(-\mathbf{v_{fwd}}),\mathbf{h}^{\perp},c_{fr},c_{fr}\cdot c_{hull})}_{\text{friction}} + d\Big(\underbrace{c_{up}c_{sail}|\mathbf{w}|^2\mathbf{s}^{\perp L}}_{\text{buoyant}} + \underbrace{c_{drift}c_{sail}|\mathbf{w}|^2}_{\text{drift}},\mathbf{h},c_{fr}\cdot c_{hull},c_{fr}\Big)\Big)$$

$$a_{rot} := \frac{-1}{c_m}\underbrace{c_{rot}|v_{rot}|v_{rot}}_{\text{friction}} + \underbrace{c_r\langle\mathbf{h},\mathbf{v}_{fwd}\rangle^2\sin(-r)}_{\text{rudder}} - \underbrace{c_{drift}c_{sail}|\mathbf{w}|\langle\mathbf{w},\mathbf{s}^{\perp}\rangle}_{\text{wind rotation}}$$

We have determined all parameters empirically by selecting values that yield reasonable sailing behavior, the values are listed in Fig. 8. While the model can easily be extended to include effects like currents, changing winds, etc., it is sufficiently complex to give the appearance of sailing as well as to require sophisticated planning techniques. We note that the simulation needs to be realized as a efficient constant time step simulation since the roadmap planner needs to compare actions om a variety of different situations.

6 Experimental Evaluation

From the great variety of possible planning tasks, we selected some interesting scenarios. For each scenario we randomly instantiate planning tasks by varying the global wind and the speed of additional vessels involved in the scenarios. We keep the courses of additional vessels fixed to gain independency of multi-agent aspects. We have selected the following types of scenarios (see Fig. 9 for illustration):

1. Sailing along a straight route
   Sail from $(0,0)$ to $(100,100)$ on a 100m wide route with wind from an arbitrary direction.
2. Sailing along a narrow route
   Sail from $(0,0)$ to $(110,100)$ on a bend route restricted in width to 20m with wind from an arbitrary direction.
3. Giving way to an oncoming vessel (rule depicted in Fig. 5(a))
   Sail from $(0,0)$ to $(100,0)$ with wind from a random northern direction (compass angle between 270° and 90°), avoiding an oncoming vessel.
4. Crossing a frequented channel (rule depicted in Fig. 5(b))
   Sail from $(0,50)$ to $(100,50)$ with wind from the west (compass angle 225° and 315°), passing behind the stern of two crossing vessels. The challenge is to start sailing slowly (which is not favored by the heuristic) in order to pass behind the other vessels before increasing sailing speed.

The vessel always starts with zero speed and the wind speed is $3ms^{-1}$. For a fixed amount of n active nodes we determine whether the planner is able to determine a solution and we record the execution time of the plan as well as the length of the trajectory computed. We regard planning as successful if the planner can determine corresponding to a trajectory to a position closer than 10m to the goal (the planner

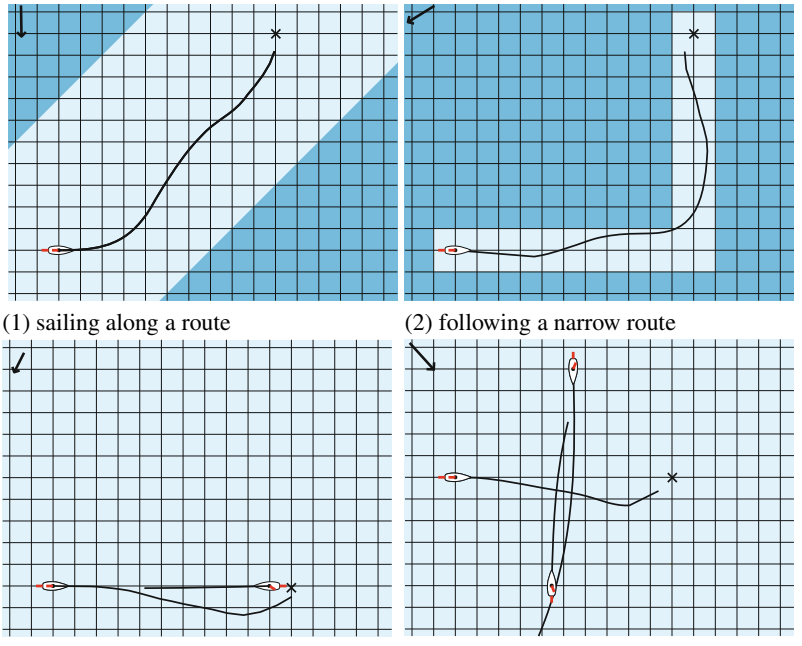


Fig. 9 Test scenarios for the evaluation including an example trajectory. Vessels are shown at their start positions, goals are marked with a cross. The grid distance is 10m.

stops immediately when a solution is found). By design, any trajectory returned is rule-compliant. For each type of scenario we randomly generate 100 instances and measures the aforementioned criteria for different choices of n. The planner has additionally been equipped with a time-out, trajectories are discarded if they exceed 500 seconds of simulation time or 150 control actions. During node expansion, 100 random actions are generated per node. The results we obtain are presented in Table 1. For every scenario we give the average over all successful planning attempts and, in parentheses, the respective standard deviation.

6.1 Discussion

Considering the success rate of planning as shown in Table 1, it can be seen that the performance varies between up to 90% for the easy scenarios to as low as 22% for the more difficult tasks, in which precise control actions are necessary. The probabilistic nature of the PRM cannot guarantee to identify a plan in all situations. However, as we could not find any systematic failure, the planner can be restarted, eventually finding a solution. Varying the amount of active nodes influences the success rate (as well as influencing the computational requirements). This can be used

Table 1 Analysis of plans obtained for the test scenarios

scenario	active nodes	success rate	path length [m]	plan duration [s]
1	20	86%	180.6 (±40.5)	64.6 (±31.1)
1	100	90%	152.2 (±16.0)	57.5 (±55.0)
1	200	93%	138.8 (±39.8)	78.0 (±72.0)
2	20	22%	201.5 (±21.9)	70.0 (±16.5)
2	100	48%	223.3 (±23.0)	68.0 (±14.8)
2	200	63%	214.8 (±26.4)	62.1 (±14.8)
3	20	81%	106.3 (±19.0)	38.5 (±8.0)
3	100	84%	105.5 (±9.2)	34.5 (±6.5)
3	200	84%	102.9 (±5.6)	31.6 (±5.5)
4	20	38%	94.2 (±12.4)	109.8 (±74.1)
4	100	66%	95.8 (± 13.9)	58.2 (±33.0)
4	200	79%	97.13 (± 16.0)	60.2 (±39.8)

to balance the need of restarts with the computational demands. For example, considering the simple scenario, it can be seen that that a set of only 20 active nodes gives a success rate of 86% which, by using ten times the amount of active nodes, can only be increased to 93%. Thus, using a small set of active nodes and restarting if necessary provides efficient means for path-planning. The measured standard deviation shows that the overall navigation quality (shortest/quickest route) leaves room for improvement, in particular the high standard deviation in the easy scenario results from some outliers in terms of long detours. However, the performance of our simple metric planning systems already indicates that a randomized-qualitative approach enables rule-compliant navigation. In particular, results for scenarios 3 and 4 show that the influence of pruning away not allowed configurations does not interfere with action planning.

As the planner does not include any pre-defined behavior, e.g., how to sail against wind or how to start sailing a vessel from a complete standstill if facing the wind, such basic sailing maneuvers have to be continuously (re-)discovered by the planner. This lack of expert knowledge can also explain the poor performance in sailing the narrow passage in scenario 2. While excluding such basic knowledge from the planning step may sound like an artificially created difficulty at first, it significantly hints at the capability of the presented approach when moving closer to a realistic sailing simulation and, ultimately, when applying the method to a real autonomous sailing vessel. With respect to a reasonable code of practice for basic sailing tasks, we believe that the abstract qualitative representation provides solid means to formalize such general rules. Our current approach can be extended to accommodate for such general rules. The key difference between right-of-way rules and rules of good practice is that the latter kind only provides default knowledge that may be violated.

7 Conclusion

This paper demonstrates how navigation rules can be formalized with qualitative constraint calculi and how qualitative reasoning can contribute to solving the navigation problem in autonomous robotic sailing. Formalizing spatial knowledge occurring in sea navigation essentially involves representation of directional spatial information, i.e., to describe the positions as seen from specific points of view (egocentric frame of reference). Our approach employs relations from the qualitative constraint calculus $\mathcal{OPRA}_m$, according qualitative reasoning methods allow us to combine information from different frames of reference into a coherent whole. Qualitative directional relations can capture static traffic regulations as imposed by buoys as well as it can capture spatio-temporal movement patterns of, for example, official right-of-way regulations or strategic maneuvers. While qualitative reasoning can be used to determine coarse, qualitative actions that are admissible with respect to the navigation rules, additional means are required to check whether such actions are possible for a specific agent in a specific physical context. In particular the kinematics of sailing vessels largely depend on the current wind, speed, etc. We use probabilistic roadmap planners to determine applicability of actions. The randomized approach to planning is particularly attractive for its ability to cover large search spaces. Furthermore, the approach can easily be integrated with a qualitative rule formalization. In this paper we demonstrate how the integration can be achieved. We also give first results of an integrated randomized-qualitative approach, demonstrating that reasonable control commands can be determined to control an autonomous robotic vessel in a rule-compliant manner.

In future work, we aim to reproduce our results in a sophisticated simulation context, stepping closer to control a real autonomous vessel. We plan to extend the qualitative rule formalization by high-level description of navigation recommendations to improve sailing performance (see [16]). While we currently use a simple model to anticipate the actions of other agents, interesting scenarios like regatta racing call for a much more involved handling of multi-agent aspects. We are confident that the qualitative rule formalization provides excellent grounds to tackle such competitive multi-agent navigation problems.

Acknowledgements. This work is supported by the Deutsche Forschungsgemeinschaft (DFG) in context of the transregional collaborative research center SFB/TR 8 *Spatial Cognition*, project R3-[Q-Shape]. Financial support is gratefully acknowledged. We also acknowledge the comments of the anonymous reviewers.

References

1. Belghith, K., Kabanza, F., Hartman, L.: Using a randomized path planner to generate 3D task demonstrations of robot operations. In: International Conference on Autonomous and Intelligent Systems (AIS), pp. 1–6 (2010)
2. Belghith, K., Kabanza, F., Hartmann, L., Nikambou, R.: Anytime dynamic path-planning with flexible probabilistic roadmaps. In: Proceedings IEEE International Conference on Robotics and Automation (ICRA), pp. 2372–2377 (2006)

3. Bhatt, M., Loke, S.: Modelling dynamic spatial systems in the situation calculus. Spatial Cognition and Computation 8(1), 86–130 (2008)
4. Cohn, A.G., Renz, J.: Qualitative spatial representation and reasoning. In: van Harmelen, F., Lifschitz, V., Porter, B. (eds.) Handbook of Knowledge Representation, pp. 551–596. Elsevier, Amsterdam (2007)
5. Dylla, F.: Qualitative spatial reasoning for navigating agents - behavior formalization with qualitative representations. In: Gottfried, B., Aghajan, H.K. (eds.) BMI Book Ambient Intelligence and Smart Environments, vol. 3, pp. 98–128. IOS Press, Amsterdam (2009)
6. Dylla, F., Frommberger, L., Wallgrün, J.O., Wolter, D., Nebel, B., Wölfl, S.: Sailaway: Formalizing navigation rules. In: Proocedings of the AISB 2007 workshop on Spatial Reasoning and Communication, pp. 470–474 (2007)
7. Freksa, C.: Conceptual neighborhood and its role in temporal and spatial reasoning. In: Singh, M.G., Travé-Massuyès, L. (eds.) Proceedings of the IMACS Workshop on Decision Support Systems and Qualitative Reasoning, pp. 181–187. Elsevier, North-Holland, Amsterdam, Amsterdam (1991)
8. Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
9. Kuipers, B.: Qualitative Reasoning: Modeling and simulation with incomplete knowledge. MIT Press, Cambridge (1994)
10. Moratz, R.: Representing relative direction as binary relation of oriented points. In: Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006), Riva del Garda, Italy (2006)
11. Pommerening, F., Wölfl, S., Westphal, M.: Right-of-way rules as use case for integrating GOLOG and qualitative reasoning. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS, vol. 5803, pp. 468–475. Springer, Heidelberg (2009)
12. Renz, J.: Qualitative spatial and temporal reasoning: Efficient algorithms for everyone. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 526–531 (2007)
13. Renz, J., Nebel, B.: Qualitative spatial reasoning using constraint calculi. In: Aiello, M., Pratt-Hartmann, I.E., van Benthem, J.F. (eds.) Handbook of Spatial Logics, pp. 161–215. Springer, Heidelberg (2007)
14. Smierzchalski, R., Michalewicz, Z.: Modeling of ship trajectory in collision situations by an evolutionary algorithm. IEEE Transactions on Evolutionary Computation 4, 227–241 (2000)
15. Statheros, T., Howells, G., Maier, K.M.: Autonomous ship collision avoidance navigation concepts, technologies and techniques. Journal of Navigation 61, 129–142 (2008)
16. Stelzer, R., Pröll, T., John, R.I.: Fuzzy logic control system for autonomous sailboats. In: FUZZ-IEEE 2007, pp. 1–6 (2007)
17. Westphal, M., Dornhege, C., Wölfl, S., Gissler, M., Nebel, B.: Guiding the generation of manipulation plans by qualitative spatial reasoning. Spatial Cognition and Computation 11(1), 75–102 (2011)

Global Data Storage for Collision Avoidance in Robotic Sailboat Racing – the World Server Approach

Nikolaus Ammann, Florian Hartmann, Philipp Jauer, Julia Krüger, Tobias Meyer, Ralf Bruder, and Alexander Schlaefer

Abstract. Collision avoidance is very important for autonomous sailing with many boats, e.g., during races. However, collision detection based on sensor data is complicated by the sails and boat motion. Particularly small boats cannot be equipped with sophisticated sensors, e.g., due to weight and power limitations. One approach to overcome this problem is to collect and store data from all participating vessels in a central data store. This World Server then provides the data to all boats, i.e., all participants in a race have access to a global view of the race situation. We present our basic server implementation and first test results indicating that the approach allows implementing and testing collision avoidance without the need for bulky and expensive sensors.

1 Motivation

In a fleet race, many sailboats compete with each other in a small area, resulting in a high chance of collisions. Human sailors have the ability of recognizing the position of each boat. Furthermore they can predict the future movements by observing the surroundings, so that they can react and avoid collisions.

In robotic sailing, however, observing the surroundings is a quite challenging task. In spite of limited sensing abilities, computing time and power supply on a robotic sailboat one has to find a way to determine where objects are, e.g. obstacles or buoys. Competitions in robotic sailing already exist [1, 2, 3], but up to now these have not included automatic collision avoidance.

So far collision avoidance in robotic sailing is done manually, although some approaches towards active object detection on vessels have been introduced [8, 9]. In contrast to large commercial vessels, where radar is a common navigation

Nikolaus Ammann · Florian Hartmann · Philipp Jauer · Julia Krüger · Tobias Meyer · Ralf Bruder · Alexander Schlaefer
Institute for Robotics and Cognitive Systems University of Lübeck
e-mail: sailing@rob.uni-luebeck.de

technology, the installation of radar systems on small boats is quite challenging. Another approach for object detection is to use cameras.

Our motivation is to develop a system that enables autonomous sailboats to be aware of their vicinity without the use of active monitoring sensors as those mentioned above. To achieve this task the boats need information such as the position and heading of other boats nearby, race course and race area. We want to ensure that each participant has access to the same data. Thus the chances of every participant in a fleet race are balanced.

Based on our experience with small boats, we decided to use a centralized system. Instead of each boat actively monitoring other objects, we introduce the so-called *World Server*. This server gathers the sensor data sent by each boat and stores it. Subsequently data from all vessels and other objects such as buoys, coastline and obstacle areas are combined into one packet and provided upon request.

In this work we present our *World Server* approach. This is motivated by an overview of two similar systems given in section 2. In section 3 we discuss potential implementations and give reasons for our choice. Subsequently, our implementation of the *World Server* is described in section 4. To conclude our paper we present some results in section 5.

2 State of the Art

In this section we shortly present two existing system for vicinity awareness. On the one hand there is the *Automatic Identification System* (AIS) which is used in commercial ship traffic. On the other hand the *RoboCup Soccer Server Simulator* is described.

2.1 Automatic Identification System

The *Automatic Identification System* (AIS) is intended for tracking ships and used by *Vessel Traffic Services* (VTS). Important goals of VTSs are [5]:

- MONITORING vessel movements,
- INFORMING mariners of other vessels and potential hazards,
- RECOMMENDING courses of action when we see a situation that the mariner may not have seen, and
- DIRECTING the outcome of situations when necessary to prevent disasters.

In order to achieve the goals mentioned above AIS provides these services with information such as unique identification, position, course, and speed of observed vessels. AIS was originally developed as a ship-to-ship navigational aid but evolved into a global standard.

Figure 1 shows an overview of the system, which works as follows [6]: The participating vessels automatically broadcast their information at regular intervals via an AIS *Very High Frequency* (VHF) Data Link transmitter. This information can be received by other vessels or stations. For example, a fleet of buoys or stations with

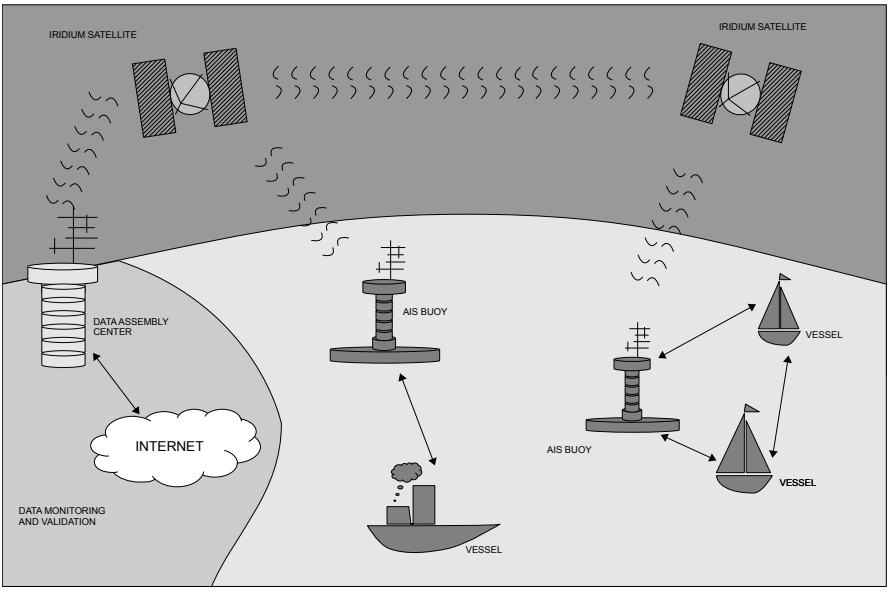


Fig. 1 This figure shows the components of the AIS: the participating vessels broadcast their information via VHF, so other vessels and stations can receive these data. The stations (AIS buoys) forward the data via Iridium satellites to the Data Assembly Centers, where the data in validated and safety relevant messages are sent back, to be broadcasted. Furthermore, the data is made available on the Internet.

low-power AIS receivers can be located offshore or along the coast. These stations provide data, such as identification, position, and other critical information from nearby vessels within radio range. The data is also forwarded via Iridium satellites to Data Assembly Centers, where it is validated and safety relevant messages and warnings are distributed to the remote stations.

Since 2000 the *International Maritime Organization's* (IMO) International Convention for the *Safety of Life at Sea* (SOLAS) requires that every registered vessel over 300 gross tones and all passenger ships regardless of size have to carry an AIS. Therefore more than 40,000 ships are currently equipped with AIS.

2.2 RoboCup Soccer Server Simulator

The *RoboCup* offers various robot competitions, e.g., *RoboCup Soccer*, *RoboCup Rescue* and *RoboCup @Home*. Another contest is the *RoboCup Soccer Simulation League* where the focus is on the artificial intelligence and team strategy during virtual soccer matches without real robots. For this competition a multi-agent system was developed: the *RoboCup Soccer Server Simulator* [7].

This system enables two teams of eleven simulated autonomous robotic players and one coach per team to play soccer. Simultaneously, the server provides monitoring and logging of the matches, as well as judging the rules. For this reason the server was built as a client/server system, which is able to handle 24 clients. Every client is a single process or program and is connected over its own port to the server via UDP/IP sockets. Depending on the current situation every client receives continuously its own match information, i.e., the server provides sensor data for the 22 player-clients. Moreover, the coach-client receives the positions and movements of all players to adapt the team strategy. Every 100 ms the server receives the player-client's commands and computes the new match state. Furthermore the server provides the connection of several monitors to observe the match and to change the server or match properties. The *RoboCup Soccer Server Simulator* is under constant development and has been in use since 1997.

3 Methods

In this section we present requirements and criteria leading to the design of a *World Server* for robotic sailing. First, we consider the topology of the communication nodes, and second we compare different communication methods.

3.1 Topology of the Communication Nodes

Our boats do not have sufficient sensors to detect other boats and obstacles on their own, and they are also limited in resources like computing power. Therefore every boat and obstacle must communicate their existence and position to the other boats in order to be detected. Based on this information the boats have to plan a collision free path.

There are basically two options to realize the communication: broadcasting, like implemented in AIS, and a centralized approach as used for RoboCup. These two approaches are discussed below.

3.1.1 Broadcasting Approach

To implement a collision avoidance algorithm only the information from the boats in a certain radius around the own position is needed. This is a perfect scenario to use a broadcast approach. Moreover, the boats would be completely autonomous, i.e., the system works without a server managing the communication process. However, for boats with limited resources it is a great effort to send, receive and process all the broadcast messages. Furthermore, it is difficult to summarize the state of the race, or to set up a judge or race committee.

Table 1 Data rate and range of WLAN (IEEE 802.11n) and enhanced Bluetooth (IEEE 802.15.1, Free2Move, Sweden) as communication link

	WLAN	Bluetooth
Data Rate	600 Mbit/s	3 Mbit/s
Range	250m	500m

3.1.2 Centralized Approach

In contrast to broadcasting, this approach is based on collecting all data by a central server, which provides it to all boats. This would be particularly useful, if the boats maintained a communication link to an onshore computer anyway. Additionally, this setup enables a global awareness of the other boats. Because all data from the boats is forwarded to the world server, the clients obtain a complete state of the world, e.g., allowing to consider different race strategies. Moreover, it is simple to check wether all boats actually send their data. Clearly, this approach would be limited to all boats staying in the vicinity of the server. As this is no limitation for smaller boats and our typical race setup, we have opted for a centralized *World Server*.

3.2 Communication Method

Regardless of the approach, boats have to communicate. Essentially, there are two widely used wireless communication technologies we considered. First, wireless local area networks (WLAN) provide high data throughput in a limited area. Second, enhanced Bluetooth modules can cover a similar area with considerably lower bandwidth, compare Table 1. However, one important aspect in the context of small boats is power consumption, which is typically far lower for Bluetooth. Either choice would be open for teams connecting to the *World Server*, and both do not require special permissions for operation.

4 Implementation

Our implementation of a *World Server* system consists of three major parts: the boats/clients, a communication gateway, and the actual world server. It is a further development of our system introduced in [4].

4.1 World Server

Figure 2 illustrates an example setup of these components. Note that in the example the client is split into a gateway routing the communication to the *World Server*, and the actual clients written in Java or C#. Clearly, this could also be implemented in a single piece of software. The *World Server* collects data from all clients,

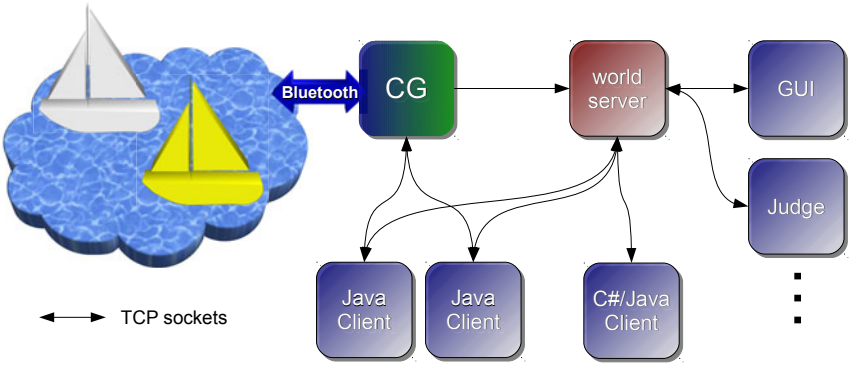


Fig. 2 *World Server* components overview with communication gateway (CG) and multiple clients. The CG handles the Bluetooth connections to the boats and directly forwards all data to the *World Server*. The server also receives data from clients not using the CG and offers all gathered information to all clients. The graphical user interface (GUI) and the not yet implemented judge connect to the *World Server* as an "ordinary" clients too.

e.g., their position, heading, or speed. This data also includes bouys and obstacles as required, and can be polled by the clients. Every object has a unique ID and a position specified by latitude and longitude (GPS) coordinates. The individual objects also carry more specific information like a radius for buoys and circular obstacles, or wind direction and wind speed for sailboats.

Our world server stores all received information in a log file for later analysis, but only keeps the most recent data for each object in the main memory and available for clients. If a client application needs a history of certain events, for example for some kind of prediction algorithm, it is up to the application itself to maintain such a history. This reduces the complexity of the data management as well as the API of the world server to a minimum. The API consists mainly of only two methods, one that retrieves a collection of all objects currently registered at the server with their up-to-date data, and another method that allows the clients to add or update their own objects.

The server offers various interfaces for the communication with the clients using different serialization methods (Java object streams, XML and a custom string-based protocol). All of these serialization methods have in common that they use TCP sockets as transport protocol.

4.2 Clients

Clients do not necessarily need to represent a physical boat; it is possible to simulate artificial traffic by adding virtual boats controlled by programs. Another example is our graphical user interface, which connects to the world server as an "ordinary"

client too. Another client could work as an artificial race judge checking for rule violations.

The world server is implemented in pure Java running on virtually any operating system. Likewise, the Java based client is platform independent. To demonstrate the flexibility of the *World Server*, we have also implemented and tested a small client written in C# using the *World Server*'s custom string-based protocol.

This protocol is based on ASCII coded strings and directly writing them to or reading them from socket streams. The first element of each message is an identifier ranging from 1 to 5 followed by the line separator '\n' or '\n\r' respectively[1]. The meanings of these identifiers are:

1 sent by the *World Server* to a client
followed by the objects registered at the server, followed by a final line separator indicating the end of the message
2 sent by a client to the *World Server*
to request the server to send all registered objects (cf. 1 above)
3 currently not used
4 sent by a client to the *World Server*
followed by an object to register/update at the server
5 sent by a client to the *World Server*
to remove an object from the server (currently not allowed)

Each object is encoded in a single line (ended by '\n' or '\n\r' respectively) following a simple scheme:

<timestamp>;TYPE:<object type>;ID:<identifier>;...
Lat:<latitude>;Lon:<longitude>;<type specific fields>\n

Timestamp[2], Type, ID, Lat and Lon are mandatory for every object, but all type specific fields are optional and will be set to 0 by the *World Server*, if not given. Parsing objects can be done easily by handling each line separately, and first splitting the string at each ';' and then at ':'[3].

The following C# code (the Java code looks nearly the same) demonstrates the easy usage of our client. It connects to the *World Server*, sends a `SailBoat` object to the server, then gets all objects from the server, and disconnects.

[1] Both options are allowed.
[2] The difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
[3] Look at our source code available on www.wrsc2011.org to see how this is done.

```
    // Create a new WorldServerClient to connect to the
    // World Server running on the localhost using the
    // default port
    WorldServerClient client
                = new WorldServerClient("127.0.0.1");

// Try to connect
if (client.connect())
{
    // Create a SailBoat object representing our boat
    SailBoat boat
            = new SailBoat("myBoat", "WinningTeam");
    // set some data
    boat.setLatitude(53.5);
    boat.setLongitude(10.0);
    boat.setHeading(180.0);
    boat.setSpeed(2.5);
    // send data to the server
    client.updateWorldServerData(boat);
    // retrieve data from the server
    IDictionary<String, WorldServerRecord> data
                    = client.getWorldServerData();
    // do something with the data
    ...
    // disconnect
    client.disconnect();
}
```

4.3 Communication Gateway

In our setup, client and boat communication are split into two seperate modules. The communication gateway shown in Fig. 2 is conceptually not directly associated with the *World Server*, but abstracts from the details of Bluetooth. It can be configured to forward all data received from boats directly to the world server, so the world server gets the data, no matter if a client uses the *World Server* or not.

5 Results

The *World Server* was tested in simulations as well as under real race conditions. The system was successfully used in a fleet race serving four boats, multiple buoys and obstacles as well as up to five clients, one for each boat and one to visualize the world state.

Table 2 shows results for simulations with different numbers of virtual objects and clients. As illustrated, the *World Server* has no performance problems handling

Table 2 Network traffic generated by the *World Server* with different number of connected boats, clients and buoys. The boats were simulated by the communication gateway and each sent 255 Byte/s to the *World Server*. We used multiple instances of our GUI to simulate the clients, each instance polled the *World Server* two times per second.

Boats	Buoys	Clients	Downstream (kB/s)	Upstream (kB/s)
10	15	1	5	7
10	15	5	6.3	25
10	15	10	8.2	48
20	15	10	13.8	85
20	15	20	18.5	160

a large number of boats and clients as the traffic stays low. Additionally our tests[4] have shown that CPU load is no limiting factor as it always stayed below 15% even with more than 100 objects.

6 Conclusion

By introducing the *World Server* we have presented a data storage system that enables all participants in a race to have access to a global view of the race situation. This system can be seen as a basis for further developments in autonomous sailing. Due to global data access for all clients it is easily possible to implement collision avoidance without active sensors. Furthermore, a central judging and race controlling system can be developed. In addition, the logging of data during a race generates a database of authentic boat movements, that, e.g., allows to test new algorithms by replaying the records in a simulator.

References

1. The microtransat, http://www.microtransat.org/ (cited May 22, 2011)
2. The sailbot competition, http://www.usna.edu/Users/naome/phmiller/SailBot/SailBot.htm (cited May 22, 2011)
3. The world robotic sailing championship, http://www.wrsc2011.org (cited May 22, 2011)
4. Ammann, N., Biemann, R., Hartmann, F., Hauft, C., Heinecke, I., Jauer, P., Krüger, J., Meyer, T., Bruder, R., Schlaefer, A.: Towards autonomous one-design sailboat racing: navigation, communication and collision avoidance. In: Proceedings of the 3rd International Robotics Sailing Conference, International Robotics Sailing Conference (June 2010)
5. VTS San Francisco, http://www.uscg.mil/d11/vtssf/ (cited May 22, 2011)

[4] Test system: Windows XP, Intel Pentium M 1.86 GHz, 2 GByte RAM.

6. National Oceanic and Atmospheric Administration, National Weather Service, National Data Buoy Center. USCG/NOAA Automatic Identification System (AIS) on Data Buoys and C-MAN Stations (2004)
7. The RoboCup Federation. User Manual RoboCup Soccer Server for Soccer Server Version 7.07 and later (2001)
8. Sauze, C., Neal, M.: A raycast approach to collision avoidance in sailing robots. In: 3rd International Robotic Sailing Conference, Kingston, Ontario, Canada, pp. 26–33 (June 2010)
9. Stelzer, R., Jafarmadar, K., Hassler, H., Charwot, R.: A reactive approach to collision avoidance in autonomous sailing. In: 3rd International Robotic Sailing Conference, Kingston, Ontario, Canada, pp. 34–40 (June 2010)

Part V
Localization and Route Planning

A Digital Interface for Imagery and Control of a Navico/Lowrance Broadband Radar

Adrian Dabrowski, Sebastian Busch, and Roland Stelzer

Abstract. The paper describes a method to establish compatibility between an autonomous surface vessel control system and a Navico Broadband Radar BR24. The solution obtains radar imagery and control of the antenna unit over its standard Ethernet interface, making the proprietary controller unit optional. It presents devices, software and methods used for empirical protocol analysis and documents the findings. Protocol details for the following functions have been identified: Operation, zoom level, various filter settings, scan speed and keep alive. An open source implementation with basic operational functionality has been made available. It features a live network mode and a replay mode using captured network traffic. In live mode, controlling radar operation as well as zoom level is possible. In both modes the radar imagery stream is rendered and displayed.

1 Introduction

Radar is an important sensor in nautical navigation, as it is also for autonomous robots. Typically a radar systems vendor delivers an integrated solution: A radar transceiver together with a display unit. Radar provides perception of the

Adrian Dabrowski
Faculty member of Höhere Technische Bundeslehr- und Versuchsanstalt Wien V - HTL Spengergasse
e-mail: dabrowski@spengergasse.at

Sebastian Busch · Adrian Dabrowski
Austrian Society for Innovative Computer Sciences
e-mail: mail@sebastianbusch.at, adrian@innoc.at

Roland Stelzer
Head of Austrian Society for Innovative Computer Sciences
e-mail: roland@innoc.at

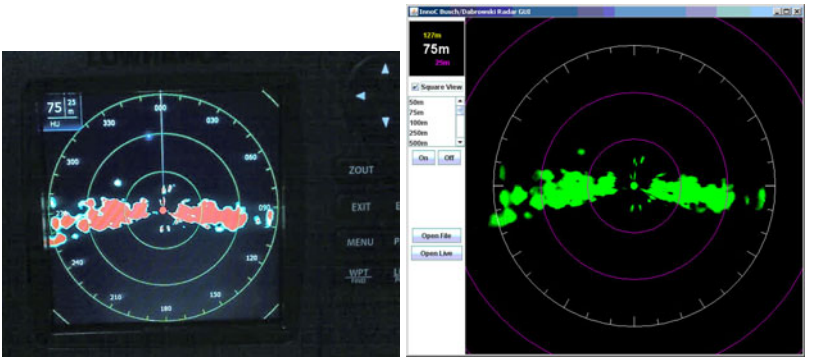


Fig. 1 A photograph of the display unit and a screenshot of the JAVA program showing the same data

surroundings with a birds-eye view, accurate range measurement and navigability under poor visual conditions.

2 Motivation

The ASV Roboat[2] needs a radar solution which provides the control algorithms with dynamic information about the surrounding real world in order to implement obstacle avoidance and possibly to handle other navigational and mapping tasks. The solution should preferably deliver its data directly in digital form.

The developers of project ROAZ II[1] as well as INNOC[2] independently contacted several vendors of small radar systems but had no success in getting interface specifications, even when offered to sign an NDA[3]. Some vendors have their own PC/Windows software on sale, others presumably want to keep that option for the future.

Russel Technologies Inc. (former Xenex Innovations Ltd) sells a commercial PC/Windows based solution [8] including an SDK for Furuno Radars. The System consists of an embedded PC which digitizes the proprietary Furuno video signal, as well as a USB-to-RS422 bridge to control the antenna unit. This system is used by the SSC San Diego USV team [6].

Within the project ROAZ II [1] a Furuno NavNet Vx2 radar based solution has been evaluated. As a result, the authors were unsatisfied that among other things the radar needs gain adjustments, which they could only do manually.

For the ASV Roboat the team decided for a Navico/Lowrance BR24 Broadband Radar [7]. This radar uses *Frequency Modulated Continuous Wave* (FMCW) radar technology. The decision was based on the low power consumption compared to

[1] LSA - LaboratÃşrio de Sistemas AutÃşnomos, Instituto Superior de Engenharia do Porto.
[2] Austrian Society for Innovative Computer Sciences.
[3] Non Disclosure Agreement.

conventional pulse radars, and the fact that Ethernet is used for communication in this system.

Pulse radars are typically rated in the kilowatt output power range, whereas this one has a peak power output of 100 mW nominal and a total power usage of about 20 W (at 12 V) in operation. Pulse radars measure the time of flight of a reflected electromagnetic signal. In contrast, FMCW radars send out signals of linearly increasing frequency and compare the frequency of an incoming reflection to the currently sent frequency. Knowing the rate of frequency increase allows the calculation of distance. Despite its low power consumption, FMCW radars can be turned on and off almost instantaneously without a significant warm up time - needing less than two seconds for spinning up. This feature will further help to reduce power consumption on long-term missions.

Already having an Ethernet network in place on the ASV Roboat leverages the integration of the radar into the control system. It proofed of much help in the empirical protocol analysis, that the lower transmission layers comply to an industry standard.

3 Structure of a LowRance Installation

The Lowrance installation (Figure 2) allows multiple sensors and consumers to be connected. For example additional sensors can be transducers or NMEA heading sensors. A *Radar Interface Box* is used to connect a power supply for the antenna unit and has a standard Ethernet port on the controller/display side.

4 Methods for Empirical Protocol Analysis

All described methods and research had been done solely to establish compatibility between our autonomous surface vessel control system and the radar system mentioned above. Neither the data nor the protocol incorporated any recognizable protection scheme.

To be able to record the data exchanged between antenna unit and controller unit we decided to use an Ethernet hub (a switch with a mirror port would work as well)

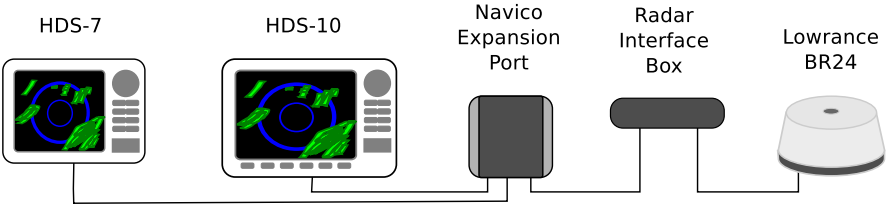


Fig. 2 A typical installation as provided by the manufacturer [7]. The optional *Navico Expansion Port* is a common Ethernet switch.

which we placed in between *Radar Interface Box* and the *Display and Control Unit*. We connected a notebook to the hub and used Wireshark[4] to record and analyse the network traffic[5].

The same setup has been used to test our implementation of the display and control functions as we could simultaneously use the proprietary controller and display and our application to compare the results.

To test which data streams are essential and which not, a computer with two Ethernet interfaces and a software bridging solution[6] has been used. It allowed us to filter out specific traffic based on layer 3 and 4 attributes without breaking the broadcast domain.

5 Network Communication

During boot-up the radar antenna transceiver and the display unit automatically configure their IP addresses. If no DHCP server is available, both devices select an IPv4 *Zeroconf* link local address as described in RFC3927[4]. All communication is done using UDP multicast datagrams. All units emit *Internet Group Management Protocol* (IGMPv3[5]) subscription announcements. This approach has two advantages. (1) It enables the system to handle multiple subscribers (i.e. controller/display units) and (2) The IP addresses of the devices are irrelevant, since they solely communicate through fixed multicast groups.

6 Control and Data Streams

Figure 3 shows the 11 data streams we identified on both devices. A *source arrow* indicates a device that actually sends data to that multicast group. A *sink arrow* indicates a device that uses IGMP to subscribe to that multicast group. Data streams that have no sink are probably used to communicate with extensions we did not install. Streams where the same device class is source as well as sink, are presumably for internal coordination between multiple devices of the same class(e.g. multiple display units).

From the five multicast streams that connects the radar and the display unit in one or the other direction, we identified two operation critical ones and tried to name them: The *Image Data Stream* and the *Control Register Access*.

[4] Wireshark is a network protocol analyser. Project website `http://www.wireshark.org/`

[5] As it turned out any IGMP unaware switch will also work. These switches will handle multicast traffic like broadcasts and therefore retransmit it on all active ports.

[6] The bridge module (`http://sourceforge.net/projects/bridge/`) is available in the official linux kernel distribution.

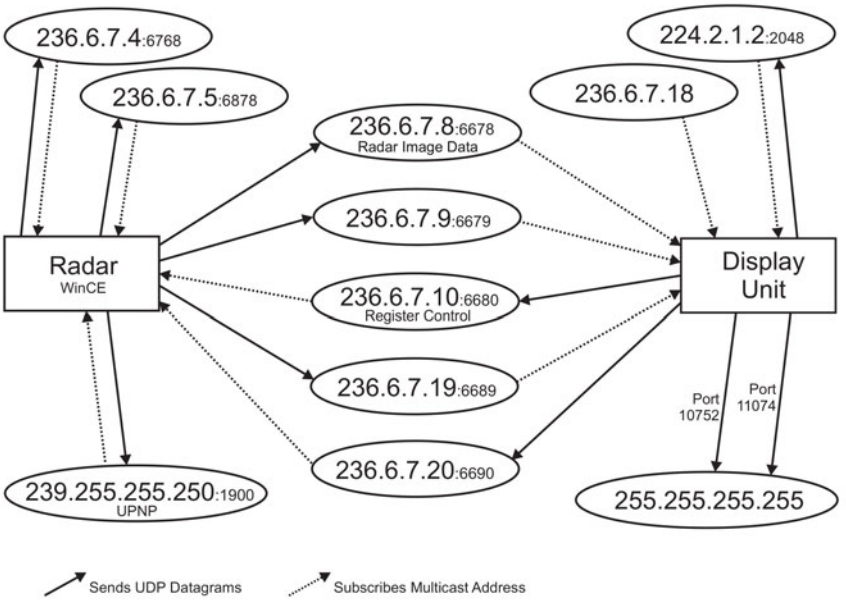


Fig. 3 Reconstructed multicast streams of a typical operation session. Port numbers are derived from actual sent data, because IGMP multicast subscriptions do not carry port information.

6.1 Radar Image Data Stream

This stream provides radar scanline data as soon as the radar is turned on by setting the appropriate control registers 0x01 and 0x02. One UDP frame containing 32 scanlines is 17160 bytes long, which get fragmented by Ethernet into 1500 byte pieces (default Ethernet MTU). For payload data structure see table 1. Multi byte fields are transmitted in little endian.

The resolution of a full 360^o scan (see figure 4) is 512 pixels by approx. 2048 scanlines as field a is incremented by two. The radar sometimes skips single or

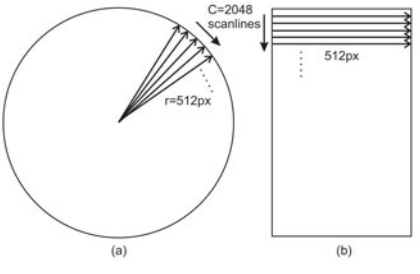


Fig. 4 Radar imagery in Cartesian form (a) and in its original polar form (b)

Table 1 Image data packet data structure

0	1	2	3	4	5	6	7	
01	00	00	00	00	20 ns	00	02 ss	Frame Header
18 l	02 st	73 rc	0B	00	44	0D	0E	Scanline Header
1D a	07	34	92	0C scale	00	00	01	
00	44	59 u1	13	00	40	00	CB u2	
512 polar pixels in 8 bit grayscale ⋮								Scanline

$\times 32$

ns	Number of scan lines, always 32d (untested)
ss	Size of scan line, always 512d (untested)
l	length of scan line header, always 24d (untested)
st	unknown, probably status: 02 valid data; 18 spin up
rc	raw scan line counter 0..4095
a	absolute angle of scan line $0..4095 = 0..360^o$
scale	scan radius in meters $= scale * 10/\sqrt{2}$
u*x*	variable, but unknown

multiple scanlines presumable when the automatic interference rejection discards them or the unit needs to re-sync mechanically.

Decoding the imagery is stateless, as all necessary information is included in the scan line frame. See Alg. 1 on how the variables are used.

Algorithm 1 Map scanline to Cartesian pixels (a, $polarpixels[512]$)

Require: $Output.width = Output.height$
  $cosa \leftarrow cos(a \cdot 360/4096)$
  $sina \leftarrow sin(a \cdot 360/4096)$
  $oscale \leftarrow Output.width/1024$
  $omid \leftarrow Output.width/2$
  for $r \leftarrow 0..511$ **do**
    $Output.pixels2D[omid + cosa \cdot r \cdot oscale, omid + sina \cdot r \cdot oscale] \leftarrow polarpixels[r]$
  end for

6.1.1 Wireless Network Transmission Issues

Although the data stream uses just around 1 MBit of net bandwidth, it is not suitable for standard WiFi (e.g. IEEE 802.11 b/g) because of the way it is send out. Access-points negotiate a separate data rate for each client depending on radio conditions. Broadcasts and multicasts however are transmitted using the lowest possible speed, so that all station in range are able to receive it. That speed is usually set to

1 or 2 MBit (gross speed) by the manufacturer and can be overwritten only in some products. The achievable net speed under good conditions is often approximated by dividing the gross speed by half. In either case the image data will use almost all airtime or even more. Lost data is inevitable under such conditions. Raising the multicast speed (where possible) on the other hand will prevent clients from associating with that access-point in greater distance or bad radio conditions.

As a consequence for the ASV Roboat, we either need an access-point that filters multicast or we need to split up the network into an internal and an external one. The latter to be accessible via WiFi for monitoring purposes.

6.2 Register Control

Multicast address 236.6.7.10 port 6680 is used to set and read control registers in the radar. Return messages seem to be sent via 236.6.7.9 port 6679 but as they are not needed for the basic operation, we have not investigated the return channel so far. An example can be found in table 2.

Some operations require to set multiple registers, in which case multiple UDP-datagrams are sent.

The names of the registers below are arbitrary. We tried to match them with their apparent function.

6.2.1 Register 01/02: Radar Operation

Payload length: 1 byte

To turn the radar on, register 0x00 and 0x01 need to be set to 1. This starts the *image data stream*. Likewise to turn the radar off, both registers have to be set to 0.

Sometimes the first scan lines after a power on may contain wrong angle information. This resolves within one full turn of the transmitter.

6.2.2 Register 03: Zoom Level

Payload length: 4 bytes (1 dword)

The register 0x03 controls the range of radar operation. The scan radius is encoded as dword in decimetres. The format suggests that any value within a valid

Table 2 Example of packet payload data for setting a register

0	1	2	3	4	5	
03	C1	10	27	00	00	Set range to 1 km
reg	cmd	data				

reg	Register number. It describes the function to access.
cmd	Always 0xC1 for *write* commands.
data	variable length payload to be written, at least one byte. Multi-byte fields are encoded *little endian*.

range can be set. We suggest to stick to the predefined ranges (see table 3) as used by the manufacturers display unit, as we can not confirm if other values produce accurate data.

A change is reflected in the *scan line header* in the radar image data stream in typically under a second.

6.2.3 Register 06: Filters and Preprocessing

Payload length: 1 byte selector + 8 byte data = 9 bytes

Register 0x06 offers access to a wide range of filters and preprocessing functions, such as *Sea Clutter Compensation*, *Rain Clutter* and *Gain control*. The first byte of the payload acts as selector, a comprehensive description can be found in table 4.

Note: There is no *automatic rain clutter* adjustment: $x2$ is in the range from 0x01 to 0x50, with 0x4D being the default value.

6.2.4 Register 08: Interference Rejection

Payload length: 1 byte

This option at register 0x08 accepts four values, where 0=*off*, 1=*low*, 2=*medium* and 3=*high*.

6.2.5 Register 0A: Target Boost

Payload length: 1 byte

The *Target Boost* setting at register 0x0A accepts three values, where 0=*off*, 1=*low* and 2=*high*.

6.2.6 Register 0E: Local Interference Filter

Payload length: 1 byte

This option has four values, where 0=*off*, 1=*low*, 2=*medium* and 3=*high*.

Table 3 Register values for zoom ranges

Range	Hex Data
50 m	0xf4,0x01,0,0
75 m	0xee,0x02,0,0
100 m	0xee,0x03,0,0
250 m	0xc4,0x09,0,0
500 m	0x88,0x13,0,0
750 m	0x4c,0x1d,0,0
1 km	0x10,0x27,0,0
1.5 km	0x98,0x3a,0,0

Range	Hex Data
2 km	0x20,0x4e,0,0
3 km	0x30,0x75,0,0
4 km	0x40,0x9c,0,0
6 km	0x60,0xea,0,0
8 km	0x80,0x38,1,0
12 km	0xc0,0xd4,1,0
16 km	0x00,0x71,2,0
24 km	0x80,0xa9,3,0

Table 4 Register values for filters and preprocessing

Function	Register Values										
	0	1	2	3	4	5	6	7	8	9	10
Automatic Gain	06 reg	C1 cmd	00 sel	00	00	00	01	00	00	00	A1
Manual Gain	06 reg	C1 cmd	00 sel	00	00	00	00	00	00	00	x1 val
Rain Clutter Filter	06 reg	C1 cmd	04 sel	00	00	00	00	00	00	00	x2 val
Sea Clutter Harbour Automatic	06 reg	C1 cmd	02 sel	00	00	00	01	00	00	00	D3
Sea Clutter Manual	06 reg	C1 cmd	02 sel	00	00	00	00	00	00	00	x3 val

6.2.7 Register 0F: Scan Speed

Payload length: 1 byte

Setting 1 increases the scanning speed, while 0 resets to normal speed.

6.2.8 Register A0: Keep Alive

Payload length: 1 byte

The radar turns off automatically after 20-60 seconds when it loses contact to the display unit. To prevent this a keep alive timer has to be reset regularly. Register 0xA0 has to be set to value of 2 every approx. 10 seconds.

7 Sample Data Acquisition Setup

By May 2011, the mounting brackets for the ASV Roboat are still in construction. To acquire real-world sample data in a dry test, we chose to mount a test setup of the system on a bike trailer. The setup (see figure 5) consists of radar antenna, radar interface box, a 12 V lead battery, radar controller unit and a laptop. For ease of use the laptop has been mounted on the front carrier of the bike and connected to the other components on the trailer. On the laptop is run: (1) the authors' JAVA implementation to control the radar and check the output and (2) Wireshark to capture network traffic for later analysis. To acquire positions, a smart phone running a GPS tracking software has been mounted alongside the radar antenna.

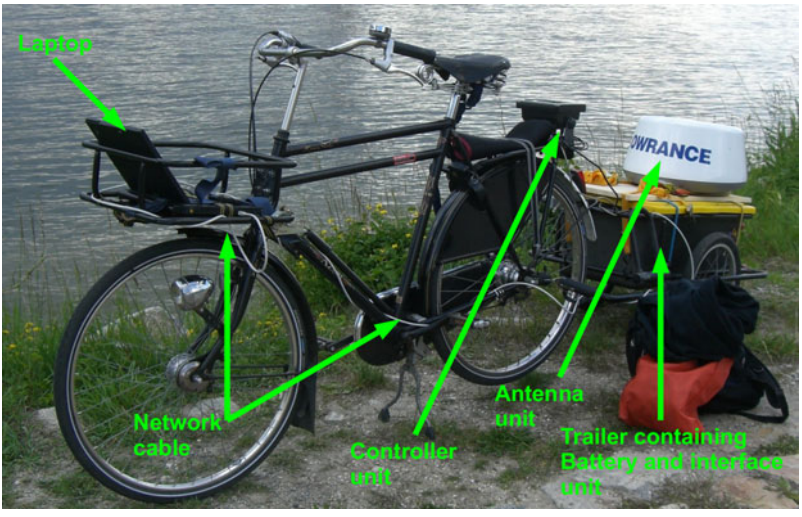


Fig. 5 The sample data acquisition setup

This setup allowed the authors to record sample data for dynamic values of speed and direction. For further analysis radar data will be mapped to the recorded GPS data and fed into the mapping module of the Roboat software to be used as sample data for steering and obstacle avoidance algorithms. In figures 6, 7 comparisons of photo and radar data give an idea of how well the setup might detect objects in fine weather conditions.

8 Power Consumption

The electronics in the radar antenna unit has no power switch. It gets supplied and boots up as soon as there is power at the radar interface box, but does not turn on the transmitter. In this *standby* mode the unit consumes 0.15 A at 12 V. In full operation

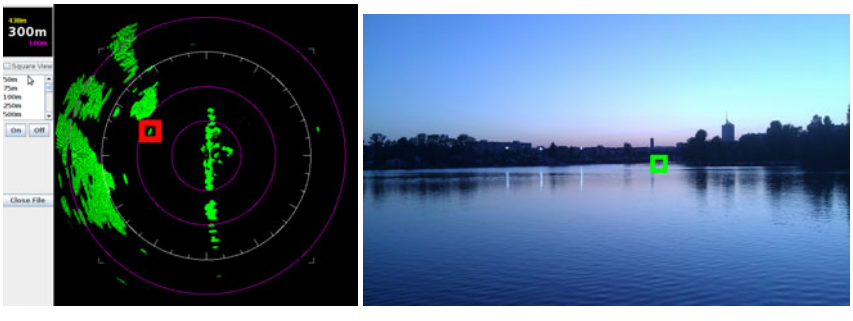


Fig. 6 Radar data/photo comparison "Alte Donau" with canoeist (radar port side)

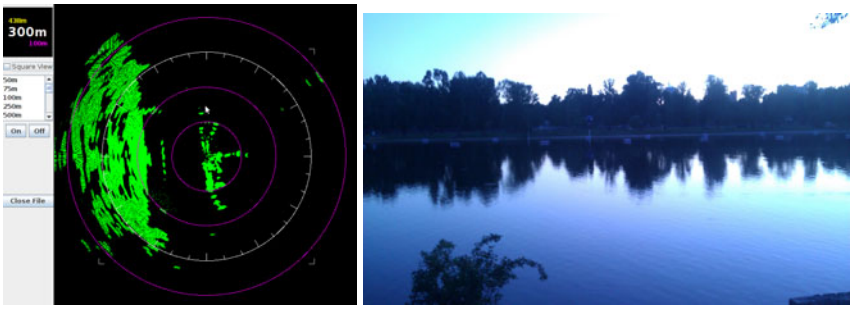


Fig. 7 Radar data/photo comparison "Alte Donau" with buoys (radar port side)

the unit consumes about 1.4 A. The display unit has a power switch and consumes about 0.7 A when turned on.

Values may vary depending on mode of operation or settings.

9 Sample Implementation

Simultaneously with the paper an open source sample implementation in JAVA is released and available at `http://www.roboat.at/technologie/radar/` It provides the following features:

Source selection	operate on live radar network connection or on a pre-recorded PCAP-File[7]
Activation	turn the transceiver on and off
Range selection	set the zoom level which is also reflected in acquisition parameters
Image decode	image data is decoded and displayed in Cartesian form.
Filter settings	may be adjusted over the network.

10 Data Usage and Future Work

Test data suggests that a strong preprocessing is necessary as radar artefacts, interference and reflections can disturb the imagery.

As discussed, image data from the radar is presented in polar pixels relative to the vessel. This makes it easy use it in a vessel steering algorithms which use the same coordinate system.

The reactive approach by Sauze and Neal [9] can be applied directly on the data, as each radar scan line can be used instead of a *raycast*.

For the radar to be used in the sweep line algorithm by Stelzer et.al. [10] radar data first has to be converted into objects representing the outline of identified

[7] The native packet capture file format of libpcap and WinPcap, which Wireshark and other tools are built upon.

obstacles (i.e. polygons in a Cartesian-like map). Together with the ship's sailing wind capabilities, transponder obstacles (e.g. AIS) and other map data this is used to generate a polar diagram of preferable directions to go [10, Fig.7]. In that diagram every obstacle creates a dent, as it makes that direction less attractive. It is the base for decisions of the *Short Course Routing* layer.

To facilitate this, the mapping module is currently extended to accept and deliver data in different formats and coordinate systems transparently hiding the conversion process. Formats are Cartesian coordinates relative to map origin and polar coordinates relative to the vessels true heading and position. That way the head-up radar data is transformed north-up. As position and heading information is only available at much larger intervals than the radar scan lines, position and heading will have to be extrapolated based on current course over ground and turn rate.

Instead of extracting and converting obstacles from polar radar to a Cartesian map, that again produces polar data, a hybrid algorithm may be used. Map and radar may each produce a preferable directions polar diagram of their own and merge them in a final step. This approach involves less complexity but does not allow to use dynamics of obstacles and therefore calculations like the time and distance of closest approach as needed for some COLREG decisions [3, section 4.2].

11 Criticism and Conclusion

The empirical found protocol described in this paper allows the direct digital access to radar imagery on a low level from a *Navico Broadband Radar* via standard Ethernet and an UDP/IP stack. This is an easily available interface for mid-range embedded systems. This paper does not deal with any of the other issues involved in operating a radar and reading its imagery. Especially it does not deal with automatic adjusting(e.g.. gain), preprocessing (eg. removing radar artefacts) or analysing the data and extracting obstacles. All topics we will have to face in the next months. We have also not tested special cases like [6] where the boat turn rate approximates that of the radar.

Furthermore on our class of boats (a 4 meter sailing vessel) the radar will be mounted at quite a low height of about 1 m and will need a heeling compensation either mechanically or algorithmically (e.g. removing/reconstructing bad parts of scan). This may require further test with different radar antenna angels relative to reference plane(ground/sea). Therefore we expect a fairly limited range in comparison with the technically possible range.

References

1. Almeida, C., Franco, T., Ferreira, H., Martins, A., Santos, R., Almeida, J.M., Carvalho, J., Silva, E.: Radar based collision detection developments on USV ROAZ II. In: OCEANS 2009 EUROPE, pp. 1–6. IEEE, Los Alamitos (2009), http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5278238 ISBN 978-1-4244-2522-8

2. Austrian Association for Innovative Computer Science - INNOC: ASV Roboat, http://www.roboat.at
3. Benjamin, M.R., Leonard, J.J., Curcio, J.A., Newman, P.M.: A method for protocol-based collision avoidance between autonomous marine surface craft. Journal of Field Robotics (2006)
4. Cheshire, S., Aboba, B., Guttman, E.: RFC 3927 - Dynamic Configuration of IPv4 Link-Local Addresses (2005)
5. Holbrook, H., Cain, B., Haberman, B.: RFC 4604 - Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast (2006)
6. Larson, J., Bruch, M., Ebken, J.: Autonomous navigation and obstacle avoidance for unmanned surface vehicles. In: SPIE Unmanned Systems Technology VIII, Orlando, FL (2006)
7. Navico: Broadband Radar TM - The Essential Guide (2009)
8. Russel Technologies Inc.: RTI Radar Installation Maintenance Manual XIR3000C (2009), http://www.russelltechnologies.ca/downloads/RTIInstallationManualXIR3000CwithFurunoRadars.pdf
9. Sauze, C., Neal, M.: A raycast appoach to collision avoidance in sailing robots. In: International Robotic Sailing Conference 2010 Proceedings, pp. 26–33 (2010)
10. Stelzer, R., Jafarmadar, K., Hassler, H., Charwot, R.: A reactive approach to obstacle avoidance in autonomous sailing. In: International Robotic Sailing Conference 2010 Proceedings, pp. 34–40 (2010)

Route Planning for a Micro-transat Voyage

Peter Gibbons-Neff and Paul Miller

Abstract. Many design decisions for an autonomous crossing of the North Atlantic Ocean require an understanding of both the time that the vessel will need to be self-sufficient and the environmental conditions expected. Potential trans-Atlantic routes were evaluated to determine both the design conditions and selecting a route that has a relatively high probability of success. Factors included: prevailing winds, currents, ice, gales, calms, sea state, sunlight, starting date, boat characteristics and ship traffic. Two methods were used; an evaluation of numerous potential routes for initial planning to determine expected times and narrow the potential field of routes and an off-the-shelf sailing yacht route optimization program for routing immediately prior to or during the crossing. The long-term method used climatological data from Pilot Charts. A Velocity Prediction Program (VPP) was used to predict the boat's performance. Calcualted passage times included the shortest, longest and most-likely. A northern and two southern routes with similar risk were identified.

1 Introduction

Preparing for an ocean crossing involves extensive planning and preparation, regardless of the vessel type. Making a crossing of the Atlantic Ocean fully autonomously is the goal of the MicroTransat Competiton and to be successful, not only is a good boat a requirement, the route selected must have a high probability of success and the boat must be designed to meet the expected conditions for the selected route. This paper discusses the research that went into the route planning for the Naval Academy's 2011 SailBot, Spirit of Annapolis, which was designed and built by undergraduate students. The long range planning used climatological data and theoretical boat velocities to estimate possible crossing times for different routes. Various factors influencing the transits were studied to provide routes with the highest

Peter Gibbons-Neff · Paul Miller
United States Naval Academy, Annapolis, Maryland, USA
e-mail: Ofc112484@usna.edu, phmiller@usna.edu

possibility of success. Recognizing that average weather predictions are often quite different from short-term forecasting, a commercial route optimization program was used with real-time data as a test of the route planning that would occur immediately prior to departure. Route planning is used by every competent navigator prior to departure and now is commonly updated with the latest weather reports throughout the voyage. Long-term planning, that which occurs more than a few months prior to the voyage, relies primarily on average climatological data. Today's motor vessels are mostly concerned with avoiding bad weather and taking advantage of favorable currents to reduce fuel consumption. Sailing vessels additionally must take advantage of favorable winds. Cornell's *World Cruising Routes* [1] is a typical reference that provides traditional sailing routes developed over centuries of experience. These traditional routes and their seasonal timing, take advantage of prevailing winds, currents, absence of ice and minimal probabilities of storms. The start of this project reviewed the traditional routes across the North Atlantic for small sailing craft. The safest routes were: "Northern Route" - July-August - Newfoundland to Ireland or Portugal "Southern Route" - November-December - Southern Europe to the Caribbean The question became which had the higher probability of success as either was feasible for a SailBot.

2 The Vessel

Inherent in a detailed analysis of either general route was a prediction of the boat's performance. This is critical in two areas, the upwind performance, which will determine whether the boat can sail against wind and current, and the seaworthiness, whether the vessel could survive expected storms. Of the two, the first is more difficult to achieve in a two-meter boat. The *Spirit of Annapolis* (SOA), a 2 meter long SailBot designed by the students was used in this project. The boat has a displacement of approximately 50 kg and a sail area of 2.4 square meters. The size was chosen to fit within the SailBot Class Rules and ease logistics. An earlier SailBot paper by Miller and his students [4] noted that SailBots are actually larger than the vessel which currently holds the *smallest vessel to cross the Atlantic* record. SOA was designed primarily for seaworthiness, rather than speed, a departure from previous USNA SailBots. To predict her speed a velocity prediction program, PCSail, was used. VPPs use wind speed and direction as inputs, and through parametric analysis of the boat's characteristics determine the forces generated. A force and moment balance then yield the speed, heel and other factors for the combination of windspeed and relative wind direction. Using the Beaufort Force values for wind so as to correlate to the U.S. National Oceanographic and Atmospheric Administration (NOAA) Pilot Charts, SOA's predicted performance in Force 4 is shown in Table 1. Predictions were made for windspeeds ranging from Force 1 through Force 8.

Of importance, in Force 2, which was a possibility for both the northern and southern routes, SOA had a minimum predicted speed of 0.8 knots. That is barely above the 0.6 knot maximum current seen on those routes. The vessel would have a difficult time sailing against the current.

Table 1 Performance Prediction for Spirit of Annapolis in Force 4. Beta is the relative wind angle. Vmdgd is the Velocity Made Good either directly upwind (beating) or downwind (running).

beta (deg)	Vel (kt)	Heel (deg)	Vmdgd (kt)
38	2.6	28.2	2.0
40	2.7	29.0	2.1
42	2.8	29.6	2.1
44	2.9	30.2	2.1
46	2.9	30.8	2.0
48	3.0	31.2	2.0
50	3.1	31.7	2.0
55	3.2	32.6	1.9
60	3.4	31.1	1.7
65	3.4	28.0	1.5
72	3.5	23.8	1.1
80	3.5	19.4	0.6
90	3.5	14.7	0.0
100	3.4	10.9	0.6
108	3.3	8.6	1.0
120	3.1	6.7	1.5
132	3.0	6.2	2.0
144	3.0	5.7	2.4
150	3.0	5.2	2.6
160	3.0	3.9	2.9
170	3.1	2.0	3.0
175	3.1	1.0	3.1
178	3.1	0.4	3.1

3 Planning Options

Route planning took two time approaches. The first was long-range planning which relied on climatological data presented in NOAA Pilot Charts [2]. The second was short-term planning which relied on weather model predictions for 24 hours to two weeks and was available in GRIB format from the National Weather Service. Both used similar formulas, although the manual method included suggestions included in Coastal Pilots as well as cruising guides. The process began with plotting courses on a gnomonic chart. A gnomonic chart contains a projection of the earth which does not distort the continents or oceans. This allows straight rhumb lines to be plotted that cross all meridians of longitude at the same angle, giving us the shortest courses. From this chart waypoints were transposed onto the distorted, but more commonly used Mercator projection chart. On the Mercator chart the rhumb lines were converted into great circle routes.

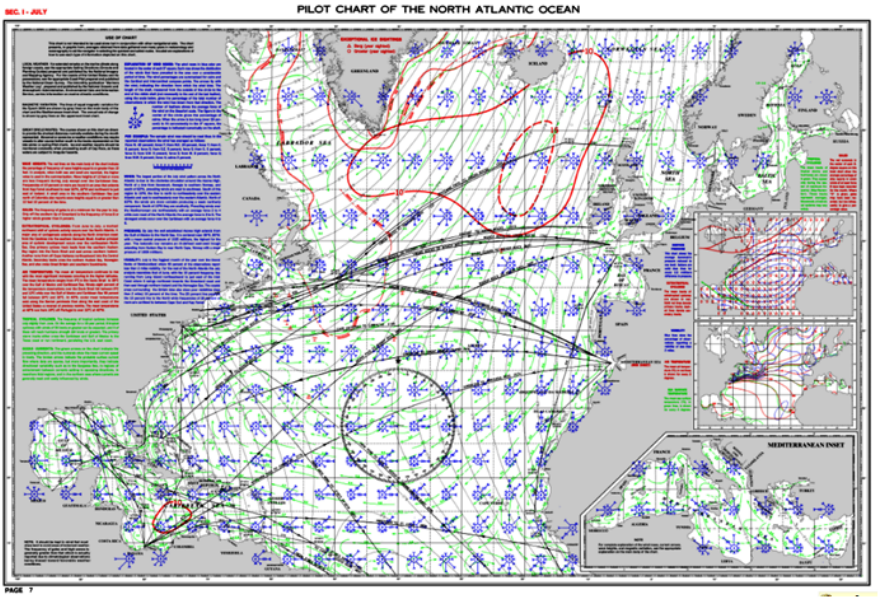


Fig. 1 NOAA Pilot Chart (NOAA) [2]

4 Long-Term Planning

A (NOAA) North Atlantic Pilot Chart (Figure 1) [2], which is a Mercator chart, was used for plotting the great circle routes and the long-term planning. There is a pilot chart for each month which presents the average of different factors necessary for preparing an ocean crossing. Each "cell" of size five degrees of latitude wide by five degrees of longitude high have a wind rose that represents the average wind force and percentage of direction from the eight major directions (north, northeast, east, etc.) averaged over more than one hundred years of data collection from vessels. Additional information includes current vectors and velocity that illustrate the prevailing currents present in that particular month. The dotted current vectors indicate variability, especially found at the surface. Sea ice is also shown, providing information such as a maximum ice limit and the mean maximum iceberg limit. The percentage of gales for every five degrees is shown, which helps determine which regions will have a greater chance for Force 8 winds and above.

Once the shortest route was plotted on the pilot chart it was compared against other routes that might provide either shorter times or higher probability of success. For instance, if an area on the great circle course showed a higher probability of headwinds or gales, a route around that area would yield a higher probability of success. The desired outcome that could be compared was the theoretical time the boat would take to cross the ocean. The data required for this calculation included the distance, direction (heading), wind (direction, percentage, and force), and

Table 2 Calculation results for Cell 1 of the shortest northern route. VMG in this table can be compared to Table 1, noting that the vessel is not designed to sail closer to the wind than 46 degrees.

Section	1					
Distance (nm)	130					
Heading	72					
True Wind Direction	%	force	Rel WA	VMG (kt)	SOG (kt)	Time (hr)
N	5	4	72	3.5	3.4	38.4
NE	4	4	27	2.0	1.9	67.4
E	4	3	18	1.7	1.6	80.4
SE	7	4	63	3.4	3.2	40.1
S	26	4	108	3.3	3.2	41.1
SW	33	4	153	3.0	2.9	44.9
W	14	4	162	3.0	2.9	44.3
NW	5	4	117	3.3	3.2	41.1
% Calm	2					
min time	38.4	hours				
likely time	44.9	hours				
max time	80.4	hours				

current (direction and velocity) for each five degree segment. Every segment was then broken up into the eight wind directions so that the time through the cell could be calculated. The process for obtaining the time to pass through the cell for each wind direction began with calculating the relative wind angle. Using north as a reference of zero degrees, the wind direction was subtracted from the heading to give the relative wind angle. Using the Beta angle from the appropriate VPP table for the wind Force gave the expected boat velocity. The beginning and ending locations in the cell gave the distance through the cell. To get the Speed of the Ground (SOG) the current vector for the cell was added to the boat speed using vector addition. The distance through the cell divided by the predicted SOG gave the time through the cell for that wind direction. In cases where the wind direction resulted in beating or running, the appropriate Velocity Made Good was used, with the condition that the boat would not sail closer to the wind than 46 degrees. From the percentage likelihood of the wind direction and strength the fastest, slowest and most-likely times through the cell were identified. Table 2 shows a typical calculation for a cell.

4.1 Possible Routes

As mentioned above, two main routes were identified. The Northern Route was from Newfoundland to Ireland and the Southern Route was from Europe to the Caribbean.

The main factors involved using the prevailing wind and current directions associated with the North Atlantic Ocean gyre and local coastal effects.

4.1.1 Northern Route

Originating in Newfoundland, a northern route would have the shortest time at sea. The island has numerous harbors that are suitable, but the main harbor of St John's or the beach at Cape Spear has the greatest potential due to its logistics, most easterly location and southerly breeze once past the headlands. Pilot Chart analysis indicates a departure time during the last two weeks of July. An earlier departure would increase the risk of pack ice and leaving later would increase the possibility of both hurricanes and extended calms. In both cases the dangerous area is the first 200 nautical miles. Figure 2 shows a detail of the Pilot Charts for July and Figure 3 shows the probability of hurricanes in the North Atlantic [5]. A more southern departure location would eliminate ice exposure but would place the vessel in heavy trafficked shipping lanes. Figure 4 illustrates the probable range of tracks and likely hood of hurricanes by month [5]. In July hurricanes are only "likely" to go as high as the Chesapeake Bay and hug the coastline. However, by August the storms are "likely" to extend up to Newfoundland and by September they can extend well toward Greenland.

While various modifications to the basic great circle route were tried, the best course from a climatology perspective will be the great circle route. The path will pass well above the Azores high and the weather should be north of the high's direct influence. However, if the high moves farther north the route could have less wind and possible regions of no wind for days [1]. From the pilot chart, the wind should be at a Force 4 for the majority of the transit, with one region showing 31% of Force 5 conditions. At the origin of the transit, the wind direction should be from the southwest. In the middle, the winds should be westerlies, and near the end of the crossing the winds should be from the west or northwest. For most of the trip the vessel will be reaching or running. Ireland has numerous potential harbors that would adequately serve as a finish location. The four finalists were Fenit, Valentia, Kinsale and Castletown-bere. Fenit has good potential due to the proximity of the Fenit Harbour & Marina to the ocean, but the entrance is still protected. It has little commercial traffic and the facilities and docks are more than sufficient for the retrieval. The 1620 nm Great Circle Route crossing from St. John's to Fenit yielded a predicted fastest time of 17.4 days, a slowest time of 30.3 days and a most-likely time of 19.4 days.

4.1.2 Southern Route

The second possible route is a southern route taking advantage of the "trade winds". The pilot charts indicate variable winds from northern Europe that can cause significant problems. While departure points in Ireland, UK and Portugal and Spain were evaluated, the only route that compared favorably to the Northern Route started in the Canary Islands, the traditional final departure point from Europe. Numerous

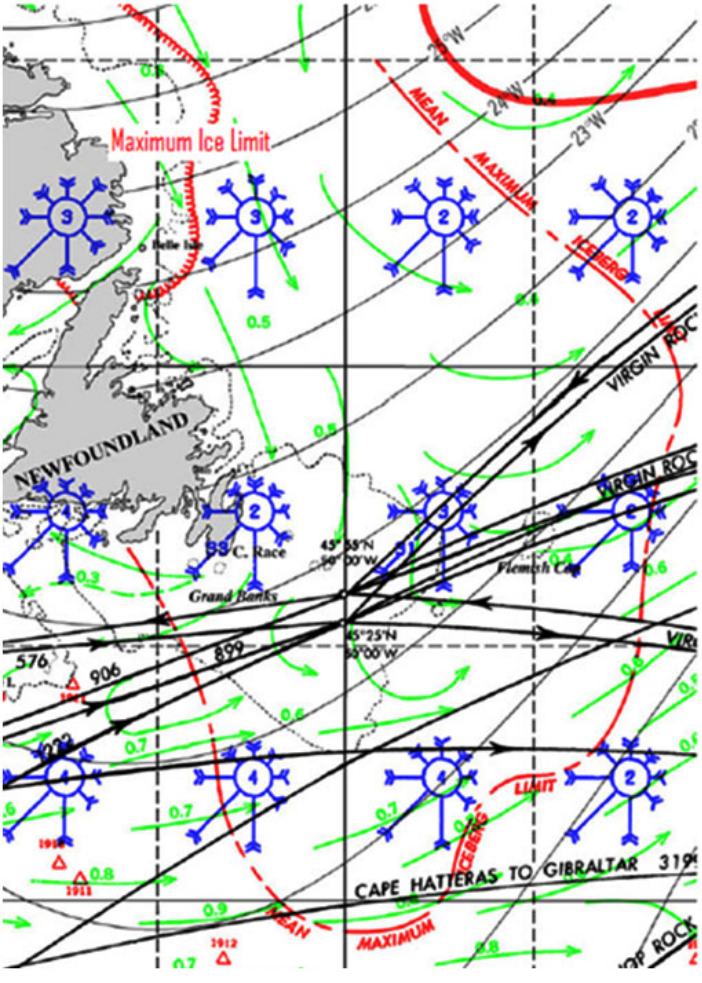


Fig. 2 Pilot Chart detail for July for the area around Newfoundland (NOAA) [2]

routes with similar transit times exist, but to minimize ship traffic the destination was selected as Antigua. This is one of the oldest routes as Christopher Columbus sailed this route during his four voyages to the Caribbean. His fastest time was 21 days [1]. The route begins from the island of Tenerife. With its larger size than the three islands to the west, it is still close to the open ocean but has a greater logistical support. Departure timing is as tight as the Northern Route. The optimal time to leave the Canaries is the last two weeks of November. Referencing Figure 3, by this time a late hurricane will be a rare occurrence. Additionally, the winter trades are seldom before the second half of November. Once the trades are initiated they blow steadily from a northeast direction; an optimal direction based on SOA's VPP data. Increasing the survivability, gale force winds are also rare in the winter months [1].

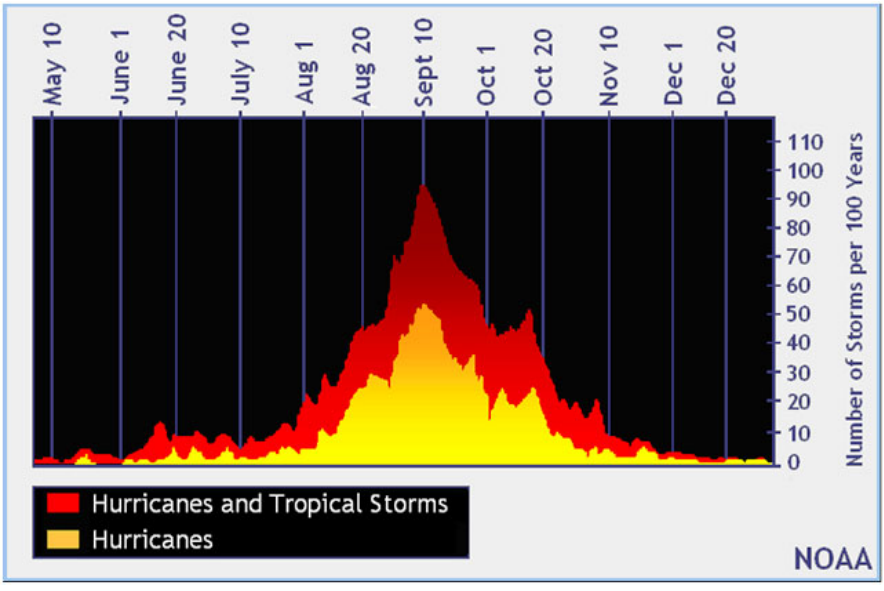


Fig. 3 Atlantic Hurricane frequencies over 100 years (NOAA) [5] showing the high probability of hurricanes in August - October

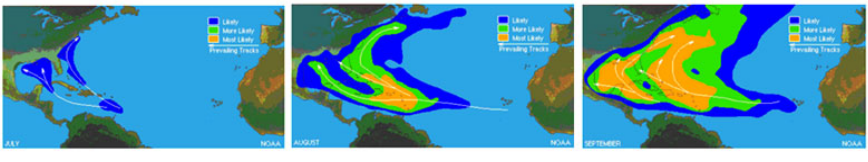


Fig. 4 Hurricane paths from July-September in the North Atlantic Ocean (NOAA) [5]

When leaving the Canary Islands the traditional objective is to leave the variable winds that surround the islands as quickly as possible. A common recommendation is to travel on a southwest heading for about 1,000 nm. This segment can possibly have northerly winds and will reach a turning point about 200nm northwest of the Cape Verde islands. From there the boat will be just south of 20°N latitude and a course change to the west is taken. This will ensure the boat reaches the northwesterly trades as soon as possible because they are rare to be found as far north as 25°N in the winter months [1]. From December to March Force 6 winds are expected. Since this is dangerous for the small SailBot, the great circle route might be optimal for survival, as long as the components and electronics could handle the chance of a longer crossing time. Current is another factor to consider. The velocity is not consistent but climatology shows an average of about 0.5 kt favorable current on both routes. This means that light air may not be that detrimental to SOA's progress in a transatlantic crossing because even if the wind is light and the boat is drifting,

it will be making progress towards its final destination. The voyage will finish off the coast of Antigua. Due to the large amount of recreational traffic and anchored vessels in each harbor the plan is to have the SailBot go in to station keeping mode a few miles off the coast until manual control is established. Using the same process as the Northern Route, the predicted times for the 2500 nm Great Circle Route Southern Route were a fastest time of 27.7 days, a slowest time of 45.4 days and a most-likely time of 30.5 days. The traditional southern route with the dog leg to avoid lighter winds is approximately 200 nm longer and could take anywhere from 27 to over 57 days with the likely time of 32 days, only a day and a half longer. Both the shorter and longer southern routes are viable.

4.2 Final Route Selection

Climatological data provided the means for determining likely times for different routes. Coastal Pilots provided information for specific departure and arrival conditions. The actual selection also included an analysis of shipping traffic. Both the Northern and Southern Routes have significant traffic, although established shipping lanes and local traffic patterns are well-documented and can be avoided. Figure 5, for instance, shows the shipping density around Newfoundland. Choosing between the Northern and Southern Routes became a question of the probability of success and mitigating potential problems. A first-order approach used the algebra of random variables based on the most likely voyage time. For instance, the reliability of the three prior USNA SailBots during sailing was calculated from testing logs as 0.9985/hr. This would imply a probability of success for the Northern Route of 58% and for the Southern Route, 33%. The uncertainty however puts this in to question as the new boat uses different, and supposedly more reliable, equipment. Additionally, while both routes have similar wind and sea conditions, the Northern Route has some potential for ice and slightly higher traffic, particularly near Ireland. Using a linear probability for ice encounter based on the extent of the berg limit versus the pack ice line and the route through the ice zone, the Northern Route drops to 49%. Ice travels slowly however and short-term forecasting can mitigate that problem. Finally, the Northern Route will provide only 50% probability of sunlight, versus nearly 100% for the Southern Route [2]. This may be significant if solar panels are the primary means of generating power.

Another approach used to evaluate the route selection was the Navy's "Green-Amber-Red" (GAR) approach. This mission risk analysis tool qualitatively evaluates each risk based on the operator's experience. The number of "Red" or high risk activities drives the "go, no-go" decision. Figure 6 shows the model applied to the two route choices. In this case the Southern Route appears less risky.

5 Short Term Planning

While the Pilot Charts offer a good understanding of the expected average conditions, most people realize that "average weather" rarely seems to occur. El Nino and

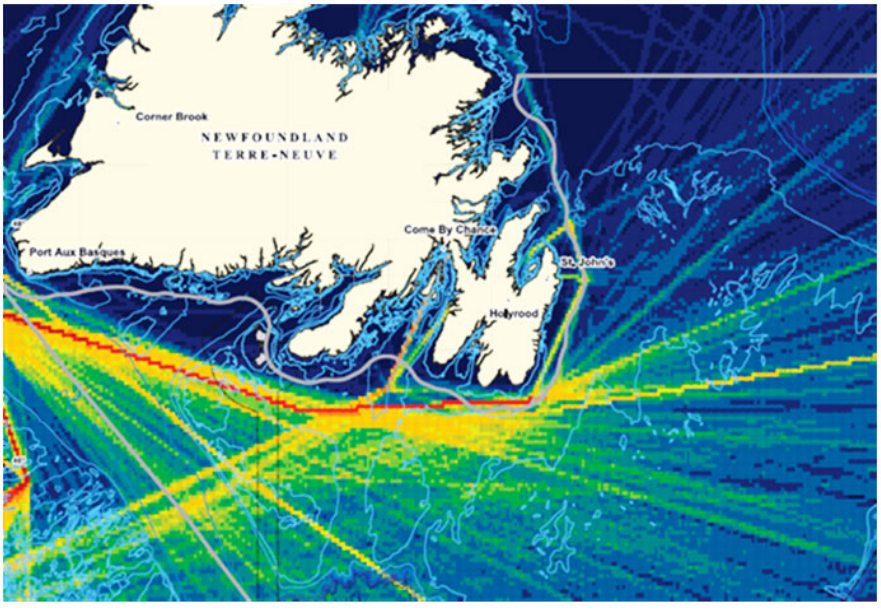


Fig. 5 Traffic density around Newfoundland indicating a northeasterly course out of St. John's will avoid the most traffic (Fisheries and Oceans Canada) [3]

Northern	Factor	Southern
	WIND STRENGTH	
	WIND DIRECTION	
	CURRENT	
	ICEBERGS	
	TIME SCALE	
	GALES	
	CALMS	
	SEA STATE	
	SHIPPING TRAFFIC	

Fig. 6 Qualitative GAR analysis of the two route options

La Nina conditions may dramatically impact average climate conditions in any given year. Short term events such as gales may dramatically differ from the long-term averages and a successful autonomous voyage will require carefully understanding the short-term forecast. Using the same approach described for the Long-Range routing, commercial software will calculate optimized courses using Gridded Binary

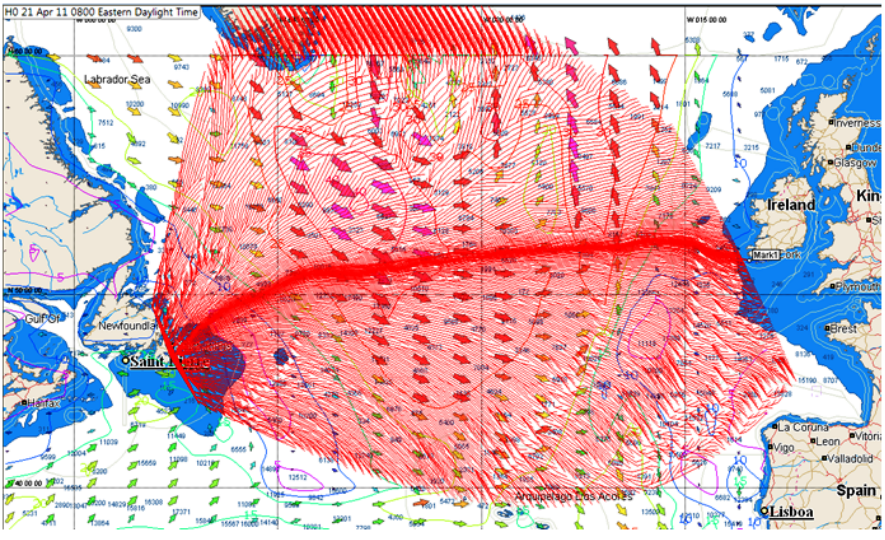


Fig. 7 Northern Route optimization for SOA using 11APR2011 GRIB file. Each red line indicates an evaluated course. The high density of courses near the rhumb line result from the optimization routine.

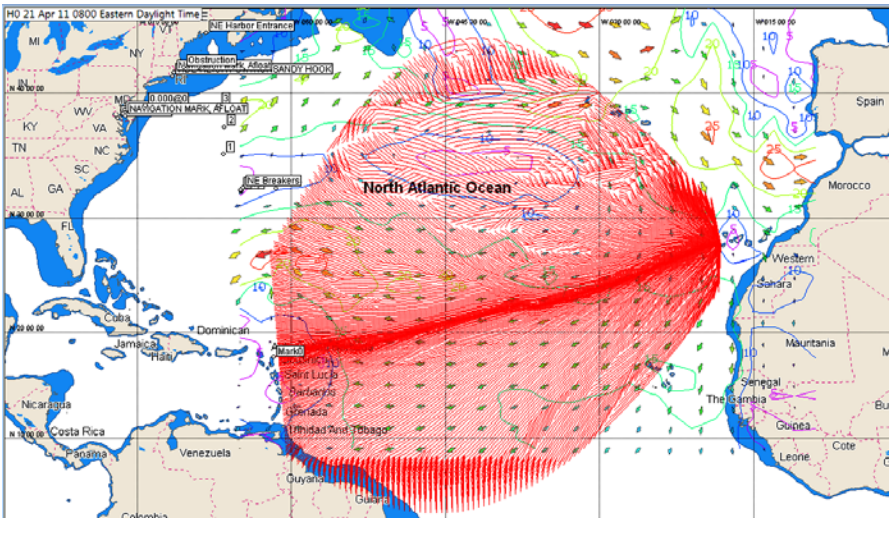


Fig. 8 Southern Route optimization for SOA using 11APR2011 GRIB file. Each red line indicates an evaluated course. The high density of courses near the rhumb line result from the optimization routine.

(GRIB) files based on short term weather models from the Global Forecast System (GFS). Multiple courses are analyzed using the vessel's VPP results and the course resulting in the shortest time is selected. The entire course can then be programmed in to the autopilot before departure. Figure 7, which was generated in Expedition for Spirit of Annapolis, shows an example from the Expedition software based on a two-week forecast for the Northern Route. Possibly due to the slow speed of SOA, the proposed route does not differ more than 75 miles from the Great Circle route. Figure 8 is a similar plot for the Southern Route. Recognizing the long duration of the voyage compared to the accuracy of two-week models, the routing can be updated onboard from satellite downloading of the latest GRIB files.

6 Conclusion

Preparing for a successful autonomous crossing of the Atlantic requires more than designing and building a good platform. A higher probability of success can be assured by choosing a route that minimizes potential weather problems. Two potential routes, each identifying a relatively high degree of success for small autonomous vessels were identified. Further research is needed to identify the specific destination for the Northern Route.

References

1. Cornell, J.: World Cruising Routes. International Marine Pub., Camden (1987)
2. Defense Mapping Agency: Atlas of pilot charts, North Atlantic Ocean. Prepared from data furnished by the Defense Mapping Agency of the Department of Defense and by the Department of Commerce (1994)
3. Fisheries and Oceans Canada, The Grand Banks of Newfoundland: Atlas of Human Activities, Commercial Shipping: Traffic Density (2000)(cited June 3, 2011)
4. Miller, P., Brooks, O., Hamlet, M.: Development of the USNA SailBots (ASV). In: 2nd International Robotic Sailing Conference, Porto, Portugal (2009)
5. National Oceanic and Atmospheric Administration's National Weather Service, 2010, Tropical Cyclone Climatology, http://www.nhc.noaa.gov/pastprofile.shtml (cited April 7, 2011)

A Rule-Based Approach to Long-Term Routing for Autonomous Sailboats

Johannes Langbein, Roland Stelzer, and Thom Frühwirth

Abstract. We present an algorithm for long-term routing of autonomous sailboats with an application to the ASV Roboat. It is based on the A*-algorithm and incorporates changing weather conditions by dynamically adapting the underlying routing graph. We implemented our algorithm in the declarative rule-based programing language Constraint Handling Rules (CHR) [4]. A comparison with existing commercial applications yields considerably shorter computation times for our implementation. It works with real-life wind forecasts, takes individual parameters of the sailboat into account, and provides a graphical user interface.

1 Introduction

Autonomous sailboats perform the complex maneuvers of sailing fully automatically and without human assistance. Starting off by calculating the best route based on weather data and going on to independent tacking and jibing, autonomous sailboats are able to sail through to any destination. Humans merely have to enter the destination coordinates.

The approach described here is planned to be implemented in the control system of the ASV Roboat, an autonomous sailing boat which has been in development by a research team of the Austrian Society for Innovative Computer Sciences (INNOC) since 2006.

So far, weather routing on the ASV Roboat relies on locally measured weather data only. This is proven to be suitable for short distances, respectively short

Johannes Langbein · Thom Frühwirth
Faculty of Engineering and Computer Science, Ulm University, Germany
e-mail: firstname.lastname@uni-ulm.de,
firstname.lastname@uni-ulm.de

Roland Stelzer
INNOC - Austrian Society for Innovative Computer Sciences, Vienna, Austria
e-mail: firstname.lastname@innoc.at

durations, such as regattas over a few miles [12]. In contrast, for long-term missions like ocean crossings, weather conditions cannot be assumed to remain stable until the boat reaches its target. Therefore, a global view with consideration of weather forecasts is necessary.

In this paper, we introduce a long-term weather routing algorithm for autonomous sailboats and show how rule-based programming facilitates a declarative and efficient implementation. In Section 2 we present how we modeled the sailboat routing problem and introduce our routing algorithm. Its implementation is then discussed in Section 3. In Section 3.1, we compare our algorithm to existing commercial solutions and discuss related work. We conclude in Section 4, which also gives an outlook on future work.

2 The Routing Algorithm

Routing for sailboats, no matter whether they are autonomous or not, can be defined as the *"procedure, where an optimum track is determined for a particular vessel on a particular run, based on expected weather, sea state and ocean currents"* [11]. In this section, we will take a closer look at the parameters required to find an optimum track and present our routing algorithm.

2.1 Modeling Long-Term Sailboat Routes

For this work, we distinguish between long-term and short-term routing in the following way: Long-term routing is the task of finding a sequence of waypoints $\mathbf{x}_0 \ldots \mathbf{x}_n$ (longitude and latitude coordinates) for a given starting point $\mathbf{x}_{start}$ and

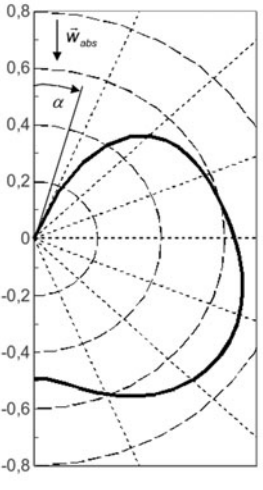


Fig. 1 The normalized polar diagram of the ASV Roboat

time t_{start} and a given destination point $\mathbf{x}_{dest}$, where $\mathbf{x}_{start} = \mathbf{x}_0$, $\mathbf{x}_{dest} = \mathbf{x}_n$ and $\mathbf{x}_k$ is reachable from $\mathbf{x}_{k-1}$ at sea while taking global weather forecasts into account. Short-term routing, in contrary, is the task of finding suitable boat headings to reach the next waypoint, given the current local weather conditions [12].

Several definitions for the term "optimum track" in the quote above are possible (see [12]), yet we want to focus on minimizing the arrival time t_{dest} at the point $\mathbf{x}_{dest}$. In order to calculate the arrival time t_k at any waypoint $\mathbf{x}_k$, we need to take weather data as well as the individual behavior of the sailboat into account.

2.1.1 Weather Data

As long-term routing is typically concerned with distances taking several days or weeks to travel, weather forecasts are required to calculate an optimal route. Weather forecasts are usually made available in the form of GRIB (Gridded Binary) files, a standardized format to store weather data [14]. In a GRIB file, wind conditions are represented as a grid of wind vectors $\mathbf{w}$, containing the wind-speed in north and east direction. A GRIB file can contain multiple forecasts, which are made available for up to 16 days in intervals as small as three hours. The resolution of the wind data typically ranges between 0.5 and 2.5 degrees.

2.1.2 Sailboat Behavior

To calculate the time required to travel between two waypoints, we need to know the speed of the sailboat for given wind conditions. This speed can be described as a function of the wind-speed and the angle between the wind and the boats heading, that is to say, the boats velocity $v = \mathrm{v}(\mathbf{w}, \alpha)$, if α denotes the true wind angle of the boat. This function is usually shown in a plot known as *polar diagram*. Figure 1 shows the normalized polar diagram of the ASV Roboat [12], which describes the relation between wind-speed and boat-speed for a given true wind angle. Another factor to take into consideration is the so-called *hull-speed*, which we treat as the approximative maximum speed of the boat. In our algorithm, this maximum speed is configurable by the user.

We approximate the travel time t_{ij} between two locations $\mathbf{x}_i$ and $\mathbf{x}_j$ on a great circle path by taking the wind conditions $\mathbf{w}_i$ and true wind angle α_i at $\mathbf{x}_i$ for the first half of the leg and the wind conditions $\mathbf{w}_j$ and true wind angle α_j at $\mathbf{x}_j$ for the second half. The distance d_{ij} between $\mathbf{x}_i$ and $\mathbf{x}_j$ is calculated using the laws of spherical geometry [2]. Together, we get the travel time $t_{ij} = \frac{1}{2} \cdot d_{ij} \cdot (v(\mathbf{w}_i, \alpha_i) + v(\mathbf{w}_j, \alpha_j))$. As sailing directly upwind is not possible, sometimes it is required to beat in order to sail from $\mathbf{x}_i$ to $\mathbf{x}_j$. We incorporate this into the travel time calculation by approximating the boats velocity made good along the great circle path between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the following way: As described in [7], we neglect the time for tacking and compute the velocity made good by using the convex hull of the polar diagram for speed calculations when sailing upwind. The true wind angle at which the boat is required to beat can be configured by the user.

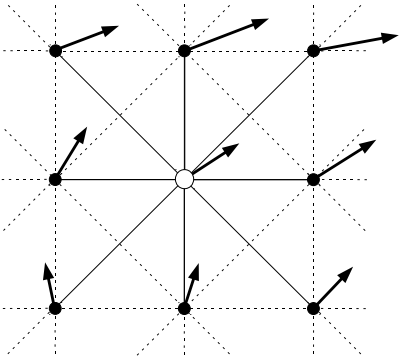


Fig. 2 An exemplary part of the routing graph. Wind vectors for each node are shown as bold arrows, edges to and from the center node as solid lines. Edges connecting the surrounding nodes are denoted as dashed lines.

2.1.3 Routing Graph

Oceans constitute a continuous search space with an infinite number of possible waypoints. To reduce the search space and make classical shortest-path methods applicable to the routing problem, we chose to discretize the search space into a grid graph with equidistant nodes, representing points on the sea. Each node is connected to its eight nearest neighbors by directed edges. Nodes located on a land mass, which are detected using a binarized world map, are not included in the graph. Nodes at locations with hazardous wind conditions, that is to say, locations, at which the wind-speed is above a user-configurable limit, are omitted as well. Figure 2 illustrates a portion of such a routing graph.

Each node in the graph is annotated with wind vectors, shown as bold arrows in the figure. In most cases, we have multiple forecasts at hand, therefore each node is annotated with one wind vector per forecast. As the location of a node usually does not coincide with a grid point in the GRIB file, bi-linear interpolation is used to calculate the wind vectors for each node. The directed edges of the graph are annotated with the travel time, calculated as shown in Section 2.1.2. Again, when multiple forecasts are present, each edge is annotated with one travel time per forecast.

The described discretization of the search space means, that the optimal route in the grid model is calculated, which is only an approximation to the true optimal route. Thus, we made the quality of the approximation configurable to the user in that the distance of the nodes in the routing graph can be arbitrarily chosen.

2.2 Calculating the Optimal Route

We chose the A*-algorithm as basis for our long-term routing as it allows the use of a heuristics for performance gain [5]. Much like Dijkstra's algorithm, the A*-algorithm assigns each node $\mathbf{x}_i$ a cost-value $c(\mathbf{x}_i)$, which is the time required to travel

to this node from the starting point. The algorithm retains an *open list* from which the currently best node is chosen for expansion. In contrary to Dijkstra's algorithm, the best node is not the node having the lowest cost but the node having the lowest value $c(\mathbf{x}_i) + h(\mathbf{x}_i)$, where $h(\mathbf{x}_i)$ is the value of the heuristics for the node $\mathbf{x}_i$. This heuristics gives an estimate for the cost to reach the node $\mathbf{x}_{dest}$ from $\mathbf{x}_i$. We use the distance from $\mathbf{x}_i$ to $\mathbf{x}_{dest}$ divided by the boats hull-speed as heuristic function $h(\mathbf{x}_i)$. It is *consistent* as it satisfies the triangle inequality $h(\mathbf{x}_i) \leq t_{ij} + h(\mathbf{x}_j)$ for two adjacent nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ and also the condition $h(\mathbf{x}_{dest}) = 0$. We choose this heuristics as it is cheap to compute and its consistency guarantees the optimality of the A*-algorithm [5].

To find an optimal route, the A*-algorithm is run on the routing graph described in Section 2.1.3. As the number of nodes and edges in the routing graph can get very high, we tried to optimize the graph construction to save memory and computation time in the following way:

Some nodes in the open list do not need to be expanded, if the destination node is reached before they are considered. Also, the fact that our heuristics is consistent guarantees that a node will never be expanded more than once by the A*-algorithm [5]. This allows to dynamically construct and deconstruct the routing graph during the execution of the algorithm:

- A node $\mathbf{x}_j$ is not created until one of the eight direct neighbors $\mathbf{x}_i$ in the grid is chosen for expansion.
- Edges are not created until two adjacent nodes are present.

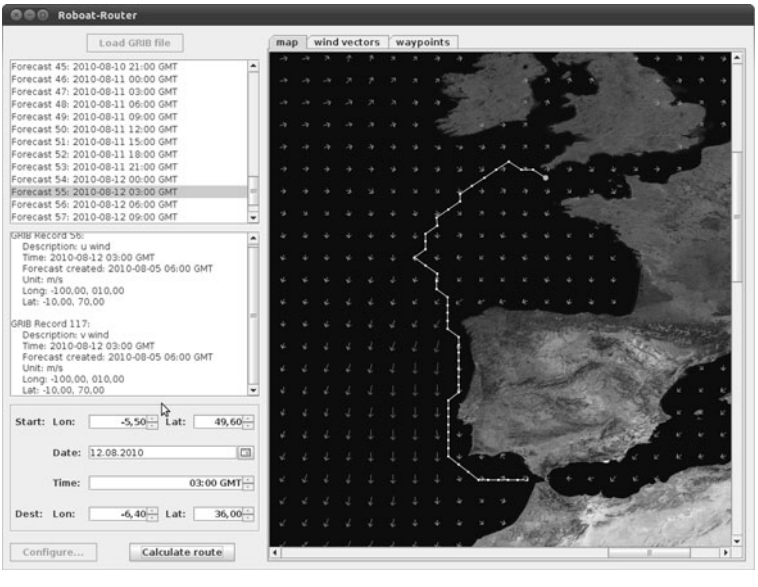


Fig. 3 The GUI showing wind conditions and the calculated route

- If a node is added to the closed list, every incoming and outgoing edge to and from this node can be removed from the graph, as nodes in the closed list will not be considered again.

The dynamic construction of the graph also facilitates another optimization: When a node $\mathbf{x}_i$ is expanded, the time $t_{start} + c(\mathbf{x}_i)$, at which this node will be reached by the boat, is known. Consequently, the forecast in the GRIB file valid at this point in time is known and the calculation of the travel time for outgoing edges of $\mathbf{x}_i$ has to be done only for this forecast and not for all forecasts in the GRIB file.

3 Implementation in CHR

We implemented our algorithm mainly in Constraint Handling Rules (CHR), combined with SWI-Prolog. For an introduction to CHR, we refer to [4]. Our implementation is based on an existing implementation of Dijkstra's algorithm with a Fibonacci heap in CHR [10], which is used as open list to achieve optimal time complexity and was extended to incorporated the use of the heuristic function. Our implementation uses CHR rules for the following tasks:

- Expanding a node: Every time a node is extracted from the open list, a rule is triggering the creation of the neighboring nodes, if they are not yet present.
- Creating edges: If there are two neighboring nodes, a rule creates an edge between the two nodes with the according travel time.
- Labeling neighbor nodes: As soon as edges are present, a rule labels the neighbors of the node expanded last with the according cost and inserts them into the open list.
- Adding node to closed list: If there are no more neighbors to be labeled, a node is added to the closed list.
- Removing edges: A node in the closed list triggers a rule removing all the incoming and outgoing edges of this node.
- Path reconstruction: Once the goal node is reached, a CHR rule reconstructs the shortest path found.

We chose CHR for its declarativity, which allowed us to implement the routing algorithm in a compact and clear fashion: The implementation in CHR consists of only 17 rules with a little under 1000 lines of code. In addition, CHR allows the routing graph to be constructed and deconstructed dynamically by simply stating the conditions, under which nodes and edges are created or removed, as rules. We furthermore believe, that the implementation in CHR facilitates future adaption and extensions in an easy way, which will help to incorporate changes that might be necessary on the ground of evaluations to come in real-life settings and conditions.

Our implementation can be used from the Prolog command line or via a Java application providing a graphical user interface. The GUI visualizes the wind conditions and allows to define starting and destination points on a world map. It also provides configuration options for the routing and displays the calculated route on the map. Figure 3 shows the GUI application.

3.1 Evaluation and Related Work

There are various commercial applications for long-term weather routing of sailboats like the *BonVoyage System*, *MaxSea*, *Sailplanner* [9], or *SailFast* [8]. The latter two are available as demo version, thus we picked them to compare our routing algorithm to. Both applications offer a GUI similar to ours. While Sailplanner automatically downloads its own wind data and is restricted to one provider (WeatherTech), SailFast allows the usage of arbitrary GRIB files. Sailplanner offers the user to pick from five different resolutions for the routing graph it uses, while SailFast uses an isochron method with an unknown resolution but with configurable time steps. However, they are both closed source applications, not revealing details about the algorithm they use for the routing.

We picked two routes of equal distance between starting point and destination (about 1440 km on a great circle path), one along the east-coast of the U.S. and one in open water to compare the running time of the two commercial solutions to our algorithm. The tests were carried out on a 2.0 GHz Intel Dual Core with 4 GB of RAM while no other applications where running and the wall clock time taken for the computation was measured. The output of Sailplanner indicates a grid width of about 30 km and 20 km for the resolutions "Medium High" and "Ultra High", respectively. Hence, we chose 30 km and 20 km as the grid resolution for the runs of our algorithm. SailFast is fixed to a 6 hour resolution for the isochron lines in the demo version. The results of the comparison are given in Table 1.

The results show, that our algorithm is considerably faster when calculating routes. A reason for this the could be the use of a heuristic function in our algorithm or the fact, that SailFast and Sailplanner seem to consider points on land in the routing as well, while our algorithm avoids them. However, the most likely reason is the fact, that more than eight different bearings for each point are considered by SailFast and Sailplanner in contrary to our algorithm. The results in Table 1 also indicate the expected trade-off between computation time and quality of the approximation to the optimal route, stemming from the complexity of the A*-algorithm which is exponential in the number of way points in the solution and thus the resolution of the grid [5].

The routes computed by our algorithm and SailFast are almost identical, while the route computed by Sailplanner is somewhat different. There are two reasons this difference might stem from: Firstly, Sailplanner uses a different polar diagram, which could not be changed in the demo version available to us. Secondly, Sailplan-

Table 1 Comparison of wall clock computation time required for routing in seconds

	SailFast 6 hours	Sailplanner "med. high"	Sailplanner "ultra high"	Roboat router 20 km	Roboat router 30 km
Route 1	109	40	254	10	6
Route 2	79	37	224	9	5

ner was run with wind data from WeatherTech, as it does not allow the import of GRIB files. Our algorithm and SailFast were both run with the same GRIB file from `saildocs.com` since the data from WeatherTech is not freely available.

In academic research, several methods have been proposed for sailboat routing (see [12] for an overview). A stochastic method for long-term routing based on dynamic programming was presented by Philpott and Allsopp [7], while methods from operations research were used by Papadakis and Perakis [6]. Recent work by the AVALON team [3] uses a routing algorithm similar to ours, however, they do not include weather forecasts in their calculations. To our knowledge, none of the aforementioned approaches have been implemented in a rule-based language like our algorithm and there has not been a published implementation of a long-term weather routing algorithm for sailboats in a declarative language.

A popular algorithm for path planning in continuous search spaces is the Theta*-algorithm [1] which also works on a grid of nodes. It allows for any-angle path planning, creating edges to all nodes in sight of the node currently expanded. An implementation of this algorithm would provide for a more accurate routing as more angles are considered. However, this would require many more nodes to be held in memory, lead to a higher calculation time, and would raise difficulties at choosing the appropriate forecasts for calculating the travel time of an edge. Hence, we chose the A*-algorithm with dynamic construction of the routing graph over Theta*.

The D*-algorithm [13] and its variants are designed for searching in dynamically changing graphs. Their advantage is the fast replanning when costs of edges are modified during the execution of the path, which is particularly interesting if a robot continuously collects new information about its environment along the route. In robotic sailing however, costs are only updated when new forecasts are available, which usually happens only every couple of hours. Furthermore, the D*-algorithm reconsiders nodes in the closed list, which would infer with the dynamic construction and deconstruction of the our routing graph (cf. Section 2.2). For those reasons, we opted against the D*-algorithm in favor of a much simpler implementation and less memory consumption, while accepting the necessity of a full re-routing once new forecasts are available.

4 Conclusion and Future Work

We proposed a long-term routing algorithm which finds an arbitrarily accurate approximation to the optimal route for a sailboat for real-life wind conditions. Our approach takes changing weather conditions into account by dynamically adapting the underlying graph used as input to the A*-algorithm. It is is implemented in Constraint Handling Rules and compares very well to existing solutions. To our knowledge, it is the first published implementation of an long-term weather routing algorithm for sailboats in a declarative or rule-based programming language.

4.1 Future Work

We use deterministic weather forecasts which are only available for a certain time ahead (see Section 2.1.1). So-called *ensemble forecasts* consist of different scenarios that can happen with given probabilities and are usually available for a longer time ahead. Research for sailing yacht races has shown how ensemble forecasts can be used to find optimal routes which perform well under all possible scenarios [7]. An extension of the routing algorithm to handle ensemble forecasts could make the routing more realistic and would allow for accurate planning of even longer routes.

Besides wind conditions, the travel time of the sailboat can be affected by currents as well. Incorporating ocean current data into the calculation of the travel time could also make the routing more accurate and may lead to better routes.

Using a graph where nodes are connected only to their eight direct neighbors puts a considerable restriction on possible paths and can lead to noticeable differences between projected and actual arrival time. This restriction could be lowered, while improving the quality of the calculated route, by using a graph where nodes are connected to 24 neighbors, as presented in [3]. We believe that this could be achieved with a reasonable increase in calculation time by replacing the CHR rules for node and edge generation.

References

1. Daniel, K., Nash, A., Koenig, S., Felner, A.: Theta*: Any-Angle Path Planning on Grids. Journal of Artificial Intelligence Research 39, 533–579 (2010)
2. Donnay, J.D.H.: Spherical Trigonometry. Interscience Publishers, Hoboken (2007)
3. Erckens, H., Büsser, G.A., Pradalier, C., Siegwart, R.Y.: Avalon: Navigation Strategy and Trajectory Following Controller for an Autonomous Sailing Vessel. IEEE Robotics & Automation magazine 17(1), 45–54 (2010)
4. Frühwirth, T.: Constraint Handling Rules. Cambridge University Press, Cambridge (2009)
5. Hart, P., Nilsson, N., Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics 4(2), 100–107 (1968)
6. Papadakis, N.A., Perakis, A.N.: Deterministic Minimal Time Vessel Routing. Operations Research 38(3), 426–438 (1990)
7. Philpott, A., Mason, A.: Optimising Yacht Routes under Uncertainty. In: Proceedings of the 15th Chesapeake Sailing Yacht Symposium, CSYS 2000 (2000)
8. SailFast LLC: SailFast Version 5.1(2011), http://www.sailfastllc.com/
9. Sailport AB: Sailplanner (2011), http://sailplanner.net/
10. Sneyers, J., Schrijvers, T., Demoen, B.: Dijkstra's Algorithm with Fibonacci Heaps: An Executable Description in CHR. In: Proceedings of the 20th Workshop on Logic Programming, WLP 2006 (2006)
11. Spaans, J.A.: Windship routeing. Journal of Wind Engineering and Industrial Aerodynamics 19, 215–250 (1985)
12. Stelzer, R., Pröll, T.: Autonomous Sailboat Navigation for Short Course Racing. Robotics and Autonomous Systems 56(7), 604–614 (2008)

13. Stentz, A.: Optimal and efficient path planning for unknown and dynamic environments. International Journal of Robotics and Automation 10(3), 89–100 (1993)
14. A Guide to the Code Form FM 92-IX Ext. GRIB Edition 1 (1994), http://www.wmo.int/pages/prog/www/WMOCodes/Guides/GRIB/GRIB1-Contents.html (accessed November 28, 2010)

Author Index

Ammann, Nikolaus 157

Becket de Megille, Francois 27
Beckmann, Daniel 71
Bishop, Bradley E. 87
Bouvart, Gabriel 55
Bradshaw, Joseph 87
Bruder, Ralf 71, 157
Busch, Sebastian 169

Clement, Benoit 27
Coquelin, Remi 27

Dabrowski, Adrian 169
De Malet, Henry 55
Douale, Nicolas 55
Dylla, Frank 141

Frühwirth, Thom 195

Gallou, Yvon 55
Gal, Oren 127
Gibbons-Neff, Peter 183

Hartmann, Florian 157
Heinig, Maximilian 71

Jafarmadar, Karim 3
Jauer, Philipp 157
Jaulin, Luc 27

Keef, Cody 87
Koch, Michael 101
Kreutzmann, Arne 141
Krüger, Julia 157

Langbein, Johannes 195
Leloup, Richard 55
Le Pivert, Frédéric 55

Meyer, Tobias 157
Miller, Paul 183

Neal, Mark 39, 113
Nicola, Jeremy 27

Petersen, Wilhelm 101

Roncin, Kostia 55

Sauzé, Colin 39, 113
Schlaefer, Alexander 71, 157
Sliwka, Jan 27
Stelzer, Roland 3, 169, 195

Taschner, Nicholas 87
Thomas, Sébastien 55

Vienney, Laurent 55

Wolter, Diedrich 141