INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
January-February 2019: 1–15

DOI: 10.1177/1729881418821830
journals.sagepub.com/home/arx
SAGE

A gain-scheduling control strategy and short-term path optimization with genetic algorithm for autonomous navigation of a sailboat robot

Davi Henrique dos Santos[1] and Luiz Marcos Garcia Goncalves[1,2]

Abstract

The development of a navigation system for autonomous robotic sailing is a particularly challenging task since the sailboat robot uses unpredictable wind forces for its propulsion besides working in a highly nonlinear and harsh environment, the water. Toward solving the problems that appear in this kind of environment, we propose a navigation system which allows the sailboat to reach any desired target points in its working environment. This navigation system consists of a low-level heading controller and a short-term path planner for situations against the wind. For the low-level heading controller, a gain-scheduling proportional-integral (GS-PI) controller is shown to better describe the nonlinearities inherent to the sailboat movement. The gain-scheduling-PI consists of a table that contains the best control parameters that are learned/defined for a particular maneuver and perform the scheduling according to each situation. The idea is to design specialized controllers which meet the specific control objectives of each application. For achieving short-term path-planned targets, a new approach for optimization of the tacking maneuvering to reach targets against the wind is also proposed. This method takes into account two tacking parameters: the side distance available for the maneuvering and the desired sailboat heading when tacking. An optimization method based on genetic algorithm is used in order to find satisfactory upwind paths. Results of various experiments verify the validity and robustness of the developed methods and navigation system.

Keywords

Unmanned surface vehicle (USV), gain-scheduling PID, sailboat robot, genetic algorithms, path optimization, nonlinear control

Date received: 18 September 2018; accepted: 5 December 2018

Topic: Robot Manipulation and Control
Topic Editor: Yangquan Chen
Associate Editor: Pengyun Chen

Introduction

The main advantage of using sailboat robots is their applicability in long-term missions, operating autonomously on water surfaces during weeks or months without a human operator. Also, it can be added the simplicity of their hardware solutions, which must be projected based on green robotics paradigms, using low-cost and low-consuming energy platforms. Conventional motorized unmanned surface vehicles (USVs) are normally propelled by fuel or

[1] Graduate Program in Electrical and Computer Engineering, Federal University of Rio Grande do Norte, Natal, Brazil
[2] Graduate Program in Computer Science, Federal Fluminense University, Niterói, Brazil

Corresponding author:
Luiz Marcos Garcia Goncalves, Department of Computer Engineering and Automation, Federal University of Rio Grande do Norte, DCA-CT, UFRN, Campus Universitario, Lagoa Nova, 59.078-970 Natal, RN, Brazil.
Email: lmarcos@dca.ufrn.br

electricity powered propeller spending the most part of their onboard energy with the propulsion system. Nonetheless, it is easy to notice that it is a challenging task, with current technology, to develop a motorized USV for long endurance missions (weeks or months) using only screw propellers powered by electrical motors, for example. There is no current technology based on naturally replenished energy, as solar cells, that can allow for this, simply because the consumption of such USV systems is bigger than any onboard power plant can produce using renewable energy. Thus, humans have to intervene or they have to be stopped after a certain time of operation, normally for refuelling or battery recharging. On the other hand, sailboats are wind-propelled, thus spending less energy than required for operation on conventional USV. This feature can be particularly interesting in long-term missions as the recent traverse of the Atlantic Ocean by the Sailbuoy Met (SB Met).[1] Other missions in environment monitoring operations, such as monitoring coral reefs, verifying quality of water, and surveillance of borders can rely on such approach, using sailboats.

There are several open problems yet to be solved related to autonomous robotic sailing. As examples, sealed cases should be used to embark electronic devices, and the sailboat hardware system must have redundant sensors in order to diminish non-systematic errors and also to keep its robustness. Also, the software architecture must be designed to have a controller that is effective, efficient, and robust, in order to allow system operation in variable situations, as the sailboat works in a highly nonlinear, and dynamic ambient that has sudden and undesirable changes, mainly in the wind parameters as its speed and direction. A simple high-level navigation system for autonomous sailboats consists basically of two parts: the path planning (high-level control) and its execution (low-level control).

The development of a low-level controller to guide the sailboat to a desired target is a relatively easy task, usually solved by using simple, with static parameters, proportional-integral-derivative (PID) heading controllers. Since modeling the sailboat dynamics in its full extension is a rather complex task, the use of PID-type controllers facilitates the tuning process, besides being of easy and fast implementation. In general, this is enough to bring the sailboat to a target using a single set of (static) control parameters (K_p, K_i, and/or K_d) throughout the entire mission, which are generally found by trial and error. In fact, such a practical experiments are reported following this simple approach in an RC Monsoon 900, with a simple proportional controller, which has been shown to work experimentally. However, further work has noticed that this approach has some critical limitations specially when the target is against the wind. Notwithstanding the fact that this strategy is easy and simple to implement, the main drawback of a PID control for sailboats is its inability to precisely approximate nonlinear behaviors, especially in tasks with specific constraints such as time and energy. In such cases, the use of a single linear control strategy is ineffective and may result in undesired behaviors. This is the main motivation for our research using GS-PI: It keeps the control strategy and development simple while increasing performance by using multiple linear controllers that are dynamically chosen depending on certain conditions to approximate nonlinear behaviors.

Another remaining challenge is to find an optimal path when the target is directly against the wind, in the area called dead zone. This angular region occurs as a fan of directions, depending on the boat construction, which is about 20° to 30° to one and to another side of the path straight against the wind direction. If a desired point is located in this region of about 40° to 60° around the direction of the wind, the sailboat cannot reach these points by following a straight-line path. This happens because in this situation, the angle of the wind on the sail does not result in a forward force component for any sail position. In conventional sailing, when the sailor (human) wants to reach points that are against the wind, a maneuver consisting of following a zigzag path is performed. This maneuver is called tacking or beating. Optimization of the tacking is an important feature for autonomous sailing once this could drastically decrease the time of the maneuvering. We noticed that the optimization of short-term trajectories for autonomous sailing is not completely solved in the literature, as it will be shown further in this text, thus being also a research focus, and one of the contributions in this article.

The contributions of our work are inside the realm of the navigation systems for autonomous sailing, more specifically we propose strategies for low-level control and short-term path planning. The first major contribution is the use of a GS-like PID control strategy, besides the use of a tuning strategy to find the parameters, which have not been used previously in the literature for autonomous sailing. PID control techniques for robotic sailboats commonly use a single set of parameters (K_p, K_i, K_d) during the whole maneuvering.[2–7] Simulation tests have shown a gain in performance mainly in narrow maneuvering conditions (water corridors) when compared to control strategies found in the literature. So we demonstrate that the performance can be enhanced when varying these gains during the maneuvering according to the wind and target directions, and we provide a way for determining the best values for these gains.

The second contribution is the use of genetic algorithm (GA) for finding the near optimal, best tacking points. There are previous techniques in the literature for finding reachable tacking points.[8–10] However, we could not find one that proposes optimization methods for autonomous sailboats to find the best tacking points with respect to several requisites such as time to target reaching, power

consumption, or traversed area. In general, these techniques use previous defined parameters, thus finding static tacking parameters that are applied in any situation. These parameters can be the number of zigzags, angle from the target direction, and apart distance from the straight original path line to the target (lateral distance).

In resume, the two main contributions (in the context of sailboat navigation automatic system) are the use of a GS-like strategy for the control and of the model-based optimization for the tacking. The main goal of this work is to improve the overall performance of the sailboat navigation system that, at first instance, consists of both the low-level controller with variant parameters and the short-term trajectory following including performing the tacking if necessary, and its optimization. The combined application of these strategies allow the sailboat to reach desirable, predefined points in the environment with a shorter time or respecting other restrictions.

Related works

There are works in the literature dealing with improved PID control algorithm, which has been applied to the control of motorized autonomous surface vehicles.[11–13] The most related to ours is the work of Larrazabal and Penas[14] in which a gain-scheduling approach utilizes PID controllers whose tuning parameters have been optimized for trajectory tracking by using a GA, for dealing with the different operation points (GS-PID-GA). That work deals with course control for trajectory tracking of a motorized USV that has active control over its linear velocity by changing the main propeller revolutions per minute (RPM). Applying a similar control strategy to sailboats is nontrivial, since its linear velocity rely on unpredictable wind forces, which translates in a reduction of maneuverability and also makes it harder to follow predefined trajectories. Thus, the main difference of our approach is the project of a control system that meets the movement restrictions of a sailboat by using less strict control objectives and tries to loosely minimize lateral distance from river banks while maintaining a general heading allowing maneuvering on narrow paths.

Related to robotic sailboat projects, we resume the several works that were found in the literature and their control strategies in Table 1. The SB Met (Sailbuoy)[1] listed in the first row of Table 1 deserves our attention because it is the first (and only) USV, to date, to complete an Atlantic crossing, achieved on 26th August, 2018. Leaving fom Newfoundland and arriving at the Ireland coast, the sailboat spent 80 days running totally autonomously at the sea. The sailboat has traveled approximately 5100 km doing kind of a tacking around the straight path to the Ireland coast, which is about 3000 km if considering to follow the (closest) straight-line planned path (on water surface). It is a 100% wind-driven unmanned remotely controlled sailboat, capable of spending up to 6 months on the water, without human intervention, with efficiency of about 50%. It has been used in long-term ocean research.[15–17] Notice that this relevant event has opened several lines of research, mainly on optimization of the performed path, which is much greater than the traveled distance in the case of a conventional (motorized) boat. The positive issue is the autonomy that the SB Met, as it does not need to use any kind of nonrenewable energy. Regarding control, the only information found is that no sail control is present leading us to understand that only the rudder is controlled, being this done remotely.[1,17] The type of controller is not reported anywhere. The Sailbuoy is self-powered, propelled through a new patent based on the wind, and is operated and controlled remotely by qualified personnel from an operations center.[17]

Table 1. Sailboat projects and their control strategies.

Projects	Rudder control	Sail control
Sailbuoy	Undefined	Undefined
Iboat	Fuzzy	Fuzzy
Roboat	Fuzzy	Fuzzy
Hyraii	LQR	Polar diagram
VAIMOS	Line following	Human behavior
Avalon	PID	Polar diagram
Aeolus	P, NL control	Based on the wind circle
ASR	PD, Sin-controller	Based on apparent wind
SailBot	PI	Based on apparent wind
FASt	PI	Polar diagram
N-Boat	PI	Based on the wind circle

LQR: linear–quadratic regulator; NL: non-linear; PD: proportional-derivative; PI: proportional-integral; PID: proportional-integral-derivative.

In Table 1, the projects Iboat,[18] Roboat[19] and[20] use a fuzzy control strategy for both the rudder and sail control differing from each other only in the pertinence functions chosen in each project. The main goal of applying fuzzy logic to sailboat control is to find rules, based on the knowledge of experienced sailors, that reproduces the sail and rudder's movement performed by a human during sailing. The Hyraii[21] project uses optimal control techniques in the form of a linear–quadratic regulator (LQR) controller. The VAIMOS[22] project uses a line follower controller, which seeks to keep the sailboat in straight lines established by a higher level navigation system. The remaining researches address the control problem by way of using PID heading controllers. The Avalon[3] project uses two PID controllers in a reference model control scheme. In its first approach, the FASt[2] project found a simple model for sailboats and derived a control law, finding the proportional and integrative parameters of its PI controller. Several projects, such as the Aeolus,[4] the Aland Sailing Robots,[5] the SailBot,[6] and the Kumar proposal[7] set the parameters of their controllers through trial and error experiments. The literature also presents some advanced control strategies applied to sailboats such as nonlinear course control[23] and line following using potential fields.[24]

Nonetheless, all of the PID controllers presented in Table 1, and their variations, use static control parameters

for the entire movement of the sailboats. Further, it is not specified any type of restriction (time to target, energy consumption, nor side distance) for choosing the right parameters. As explained above, this changing of parameters is attractive mainly because our system works in a highly nonlinear environment that needs a different strategy at each situation of wind. So, following the success of gain-scheduling control strategies in other types of mobile robots[11–14,25–29], we propose to use such approach in this work for the development of the sailboat control system. The proposed method uses experimental data obtained in simulation to decide which controller parameters satisfy specific constraints. For example, if the sailboat goal is to reach the target at the smallest amount of time, that is, the system has a time constraint, our method finds the control parameters that most satisfy such said constraint. We have investigated the hypothesis that using more than one pair of parameters (K_p, K_i) for the control improves the overall performance of the system during a specific application. So the work presented here differs from the others by proposing a model-based method to identify the best controller for a desired application through simulations. This controller, called gain-scheduling PI controller or GS-PI controller, consists of setting a table with the parameters (K_p, K_i) that produces satisfactory results for each initial heading and wind direction on the sailboat. Instead of using only a pair (K_p, K_i) for all sailing maneuverings, the system uses specific control parameters to each maneuvering, depending on the context. Simulated experiments are performed to verify the validity of the initial hypothesis, which will be shown further.

Works dealing with the upwind situation

Some few references can be found in the literature about short-term path planning and optimization strategies for sailboat navigating against the wind. The strategy adopted to solve this problem can be classified as deterministic or probabilistic. Deterministic strategies make use of locally sensed data (instantaneous) about the wind (as speed and direction) and the boat desired heading to calculate the best path. The probabilistic approach is based on statistics, using the wind probability distribution in order to determine the trajectory that is near the optimum. Some deterministic approaches found can rely on techniques such as potential fields, in which the dead zone (direction against the wind) is treated as a virtual obstacle,[10] or using the polar diagram of the sailboat, which is previously determined, performing by hand annotations of the headings that are optimum given all of the wind directions.[30] Also, there are other strategies that use fixed values for the heading, depending only on the wind direction.[8]

Differently from the above strategies that have been, all of them, validated in robotic platforms, the strategies relying on the probabilistic approach have not yet been reported on autonomous robotic sailboats. They are often used in regattas to help sailors to take the best actions given certain weather conditions. Such approaches use Markov decision processes[31,32] and Markov chains[9] for estimating the trajectory that is optimum, helping human sailors to take their decisions.

Work contextualization

The GS-PI heading controller for sailboat robots proposed here uses experimental data obtained in simulation to find near optimal variable control parameters (K_p, K_i) which meet the control objectives of the desired application. This approach differ from the above traditional PID sailboat heading controllers found in literature, which generally use a single (static) set of parameters for all maneuverings. Here we show that the use of a control table that adjusts the control parameters to each maneuvering improves the sailboat performance in each particular application. The other problem addressed here is the short-term path planning method for executing the tacking, allowing to go to targets in the dead zone. Our approach differs from the literature as it introduces a very simple and effective way for executing the tacking. Basically, information about the available area for sailing, the sailboat desired target, and the heading is used to calculate, when necessary, the tacking waypoints from the current position toward the target position. The resulting tacking trajectory is optimized using a GA in a model-based approach. This evolutionary model provides near optima multiple paths, from which the best one can be chosen. This method is implemented and tested using the simulator described next, and it is currently running in the 2.5 m sailboat N-boat II.

Sailboat dynamics and kinematics (with a 4-DOF simulator)

In order to test and verify our proposed methods, we improved a sailboat model developed by researchers of the University of Southern Denmark.[33] This simulator model describes the dynamics of a sailboat with 4-degrees of freedom (DOF), which differs of a traditional sailboat model because it also considers the roll angle. Figure 1 shows the used coordinate system and Table 2 shows the notation adopted in the current work in order to get the simulator to have a behavior as similar as possible to the real version of our robotic sailboat.

As said, this modified model is currently implemented in Matlab using the Simulink toolbox, with a dynamics and kinematics approach that is approximately (as possible) the one of our real sailboat. The general vector model that represents the sailboat movement is given by

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu}, \boldsymbol{\eta}) + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} \qquad (1)$$

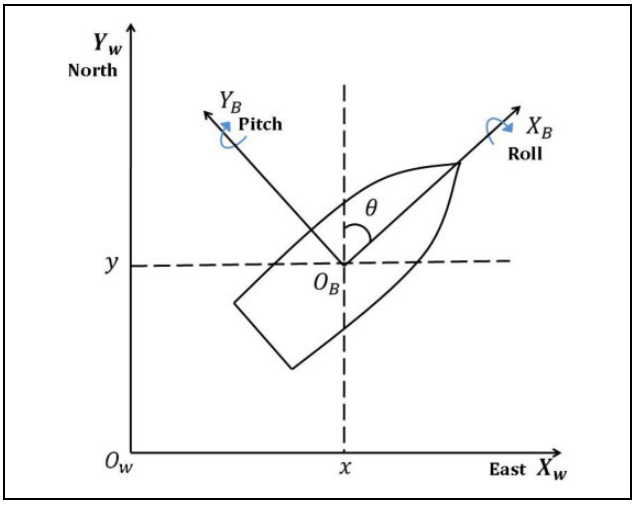

Figure 1. General coordinate system for a sailboat.

Table 2. Adopted notation.

Notation	Description
a, b, c, d	Constant coefficients to be determined empirically
a_X, b_X	Slope and y-intercept of line X
$\boldsymbol{C}, \boldsymbol{C}_{RB}, \boldsymbol{C}_A$	System/rigid-body/added-mass Coriolis-centripetal matrix
$d_{a,b}$	Euclidean distance between points a and b
d_l	Lateral distance
$\boldsymbol{D}, \boldsymbol{D}_k, \boldsymbol{D}_h$	System/keel/hull vector of damping
g	Vector of restoring forces
$\boldsymbol{I}$	Principle moment of inertia in 4-DOFs in the b-frame
$\boldsymbol{J}$	Coordinate transformation matrix
L, D	Lift and drag forces acting on the foils
m	Total mass of the sailboat
$\boldsymbol{M}, \boldsymbol{M}_{RB}, \boldsymbol{M}_A$	System/rigid-body/added-mass inertia matrix
$\boldsymbol{M}_r$	Static-righting moment
N_{ger}	Number of generation
p, r	Angular velocities on w-frame
P_0, P_d	Initial and target points
$Proj_{m,A}$	Projection of point m in line A
t_a	Time to target in seconds
u, v	Linear velocities on w-frame
x, y	Position (x, y) on the w-frame
$X/Y/K/N_{\dot{u}-\dot{r}}$	Added mass coefficients in the b-frame
α	Angle of attack on b-frame
α_{aw}	Apparent wind angle on b-frame
α_{tw}	True wind angle on w-frame
δ_s, δ_r	Sail and rudder angle on b-frame
$\boldsymbol{\eta}$	Position and orientation vector on w-frame
θ_d	Desired heading on w-frame
θ_t, d_t	Tacking angle and distance
λ	Tacking points distance constant
$\boldsymbol{\nu}$	Velocity vector in the b-frame
ρ	Flow density
$\boldsymbol{\tau}$	Vector of propulsive forces
v_a	Apparent velocity in the b-frame
ϕ, θ	Euler's angles in the w-frame

DOF: degrees of freedom.

where $\boldsymbol{M}$ (equation (2)), $\boldsymbol{M}_{RB}$ (equation (3)), and $\boldsymbol{M}_A$ (equation (4)) are the system, rigid-body, and added-mass inertia matrices, respectively. These matrices are written as

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A \tag{2}$$

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m\boldsymbol{I}_{2\times 2} & \boldsymbol{0}_{2\times 2} \\ \boldsymbol{0}_{2\times 2} & \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{I} = \begin{bmatrix} I_{xx} & -I_{xz} \\ -I_{xz} & I_{zz} \end{bmatrix} \tag{3}$$

$$\boldsymbol{M}_A = -\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{p}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{p}} & Y_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{p}} & K_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{p}} & N_{\dot{r}} \end{bmatrix} \tag{4}$$

The terms $\boldsymbol{C}(\boldsymbol{\nu})$, $\boldsymbol{C}_{RB}(\boldsymbol{\nu})$, and $\boldsymbol{C}_A(\boldsymbol{\nu})$ are the system (equation (5)), rigid-body (equation (6)), and added (equation (7)) Coriolis-centripetal matrices, respectively, which are given by

$$\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu}) \tag{5}$$

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & -\mathrm{M}_r & 0 & 0 \\ \mathrm{M}_r & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} \boldsymbol{0}_{2\times 2} & \boldsymbol{C}_{A12}(\boldsymbol{\nu}) \\ \boldsymbol{C}_{A21}(\boldsymbol{\nu}) & \boldsymbol{C}_{A22}(\boldsymbol{\nu}) \end{bmatrix} \tag{7}$$

The term $\boldsymbol{D}(\boldsymbol{\nu}, \boldsymbol{\eta})$ is the system damping matrix (equation (8)) taking into account four components: keel (equation (9)), hull (equation (10)) and the added heel and yaw damping (equation (11)). These components are given by

$$\boldsymbol{D}(\boldsymbol{\nu}, \boldsymbol{\eta}) = \boldsymbol{D}_{\mathrm{k}}(\boldsymbol{\nu}) + \boldsymbol{D}_{\mathrm{h}}(\boldsymbol{\nu}, \boldsymbol{\eta}) + \boldsymbol{D}_{\mathrm{heel}}(\boldsymbol{\nu}) + \boldsymbol{D}_{\mathrm{yaw}}(\boldsymbol{\nu}, \boldsymbol{\eta}) \tag{8}$$

$$\boldsymbol{D}_{\mathrm{k}}(\boldsymbol{\nu}) = \begin{bmatrix} -L_k \sin\alpha_{ak} + D_k \cos\alpha_{ak} \\ -L_k \cos\alpha_{ak} - D_k \sin\alpha_{ak} \\ (-L_k \cos\alpha_{ak} - D_k \sin\alpha_{ak})|z_k| \\ (L_k \cos\alpha_{ak} + D_k \sin\alpha_{ak})|x_k| \end{bmatrix} \tag{9}$$

$$\boldsymbol{D}_{\mathrm{h}}(\boldsymbol{\nu}, \boldsymbol{\eta}) = \begin{bmatrix} F_{rh}(v_{ah}) \cos\alpha_{ah} \\ -F_{rh}(v_{ah}) \sin\alpha_{ah} \cos\phi \\ (-F_{rh}(v_{ah}) \sin\alpha_{ah} \cos\phi)|z_h| \\ -F_{rh}(v_{ah}) \sin\alpha_{ah} \cos\phi|x_h| \end{bmatrix} \tag{10}$$

$$\boldsymbol{D}_{\mathrm{heel}}(\boldsymbol{\nu}) + \boldsymbol{D}_{\mathrm{yaw}}(\boldsymbol{\nu}, \boldsymbol{\eta}) = \begin{bmatrix} 0 \\ 0 \\ c\dot{\phi}|\dot{\phi}| \\ d\dot{\theta}|\dot{\theta}| \cos\phi \end{bmatrix} \tag{11}$$

The terms L and D used in equations (9), (20), and (21) are the lift and drag forces, which are respectively given by (equation (12))

$$L = \frac{1}{2}\rho A v_a^2 \, \mathrm{C}_L(\alpha), \quad D = \frac{1}{2}\rho A v_a^2 \, \mathrm{C}_\mathrm{D}(\alpha) \tag{12}$$

The term $\boldsymbol{g}(\boldsymbol{\eta})$ contains the restoring forces given by equation (13)

$$\boldsymbol{g}(\boldsymbol{\eta}) = \begin{bmatrix} 0 \\ 0 \\ a\phi^2 + b\phi \\ 0 \end{bmatrix} \tag{13}$$

The term $\boldsymbol{\eta}$ is the vector containing the position and orientation of the boat with respect to the frame W and is given by equation (14)

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{14}$$

$$\boldsymbol{J}(\boldsymbol{\eta}) = \begin{bmatrix} \boldsymbol{J}_1(\boldsymbol{\eta}) & \mathbf{0}_{2\times 2} \\ \mathbf{0}_{2\times 2} & \boldsymbol{J}_2(\boldsymbol{\eta}) \end{bmatrix} \tag{15}$$

$$\boldsymbol{J}_1(\boldsymbol{\eta}) = \begin{bmatrix} \cos\theta & -\sin\theta\cos\phi \\ \sin\theta & \cos\theta\sin\phi \end{bmatrix} \tag{16}$$

$$\boldsymbol{J}_2(\boldsymbol{\eta}) = \begin{bmatrix} 1 & 0 \\ 0 & cos\phi \end{bmatrix} \tag{17}$$

Here, the parameter $\boldsymbol{\nu}$ is the velocity vector with respect to frame B and is given by equation (18)

$$\dot{\boldsymbol{\nu}} = -\boldsymbol{M}^{-1}\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{M}^{-1}\boldsymbol{D}(\boldsymbol{\nu},\boldsymbol{\eta}) - \boldsymbol{M}^{-1}\boldsymbol{g}(\boldsymbol{\eta}) + \boldsymbol{M}^{-1}\boldsymbol{\tau}(\boldsymbol{\eta},\boldsymbol{\nu},\delta_r,\delta_s,v_{tw},\alpha_{tw}) \tag{18}$$

The current state-space representation of the sailboat is given by equations (14) and (18). Finally, $\boldsymbol{\tau}$ is the vector with the resulting propulsive forces of the sailboat given by equation (19), where $\boldsymbol{\tau}_s(\boldsymbol{\eta},\boldsymbol{\nu},\delta_s,v_{tw},\alpha_{tw})$ and $\boldsymbol{\tau}_r(\boldsymbol{\nu},\delta_r)$ are the sail and rudder resulting forces, respectively, given as

$$\boldsymbol{\tau}(\boldsymbol{\eta},\boldsymbol{\nu},\delta_r,\delta_s,v_{tw},\alpha_{tw}) = \boldsymbol{\tau}_s(\boldsymbol{\eta},\boldsymbol{\nu},\delta_s,v_{tw},\alpha_{tw}) + \boldsymbol{\tau}_r(\boldsymbol{\nu},\delta_r) \tag{19}$$

$$\boldsymbol{\tau}_s(\boldsymbol{\eta},\boldsymbol{\nu},\delta_s,v_{tw},\alpha_{tw}) = \begin{bmatrix} L_s\sin\alpha_{aw} - D_s\cos\alpha_{aw} \\ L_s\cos\alpha_{aw} + D_s\sin\alpha_{aw} \\ (L_s\cos\alpha_{aw} + D_s\sin\alpha_{aw})|z_s| \\ -(L_s\sin\alpha_{aw} - D_s\cos\alpha_{aw})x_{sm}\sin\delta_s \\ +(L_s\cos\alpha_{aw} + D_s\sin\alpha_{aw})(x_m - x_{sm}\cos\delta_s) \end{bmatrix} \tag{20}$$

$$\boldsymbol{\tau}_r(\boldsymbol{\nu},\delta_r) = \begin{bmatrix} L_r\sin\alpha_{ar} - D_r\cos\alpha_{ar} \\ L_r\cos\alpha_{ar} + D_r\sin\alpha_{ar} \\ (L_r\cos\alpha_{ar} + D_r\sin\alpha_{ar})|z_r| \\ (-L_r\cos\alpha_{ar} - D_r\sin\alpha_{ar})|x_r| \end{bmatrix} \tag{21}$$

The simulator has a state equation structure, and the sailboat state vector is given by $[x; y; \phi; \theta; u; v; p; r]$. Its processing flow is composed by a heading control, a sailboat model, itself, and by some user interface, a graphical component. As named, the heading control component is a controller for allowing the sailboat to execute some predefined trajectory. Its input is a reference direction, as for example the azimuth given by the actual position of the sailboat and a target waypoint, for which the actual sailboat direction is computed. The displacement error from this reference direction to the actual sailboat direction (sailboat orientation given by the sensors) is calculated in order to adjust the rudder orientation, with respect to the sailboat. The second block, the sailboat model, is responsible for determining the following sailboat states, based on the current state and the new calculated parameters, using the equations above introduced for defining the type of movement to be performed by the sailboat, which are also abovementioned. Finally, the sailboat and the environment are redrawn in the graphical interface that provides the sailboat visualization in three-dimensional (3-D), as seen in Figure 2. This provides to the user a way to track the behavior of the sailboat as the simulation goes on.

As said, the original simulator model developed by Xiao et al. at University of Southern Denmark (USD)[33] has been modified in our work to agree with the requirements of our sailboat project. The first modification that was necessary is to replace the original heading control by the GS-PI controller developed in this work. We also changed the original sailboat parameters to meet the specifications of our sailboat.

The proposed guidance and control system

Different types of applications imply different control objectives. For example, a water quality monitoring

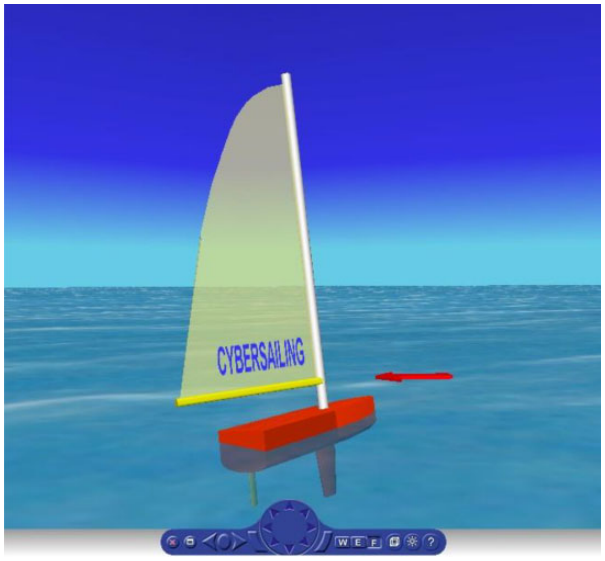

Figure 2. Visual representation of the sailboat simulator.

mission on a river may require maneuvering in small or narrow areas or respecting some power use requirements where velocity is not a relevant issue. Notice that such restrictions imply that the best path is usually not the fastest path. On the opposite, in a rescue mission, the sailboat must reach the target in the smallest possible time even if this compromises energy management. Long-term missions may require the execution of maneuverings that reduces energy consumption, preserving the robot energy autonomy with self-sufficiency. Thus, the control challenge addressed in this work is the development of a method to find GS-PI low-level controllers for each specific application.

Gain schedule PI control: GS-PI

The idea behind GS-PI is to design specialized controllers that meet the specific restrictions of each application. In this way, a higher level agent can select the best available controller according to the current maneuvering being performed, increasing system efficiency. In the sailboat, the low-level controller must act specifically on the rudder and on the sail to obtain the desired motion. Acting on the sail is simpler, the problem can be summarized in finding an angle within the sail's movement space, according to the actual wind direction given by the wind sensor. For the rudder, a PI-type heading controller is used due to its simplicity, robustness, and fast implementation. Changes in the proportional and integral parameters modify the rudder behavior and consequently affect the movement of the sailboat. The control problem is then reduced to finding the K_p and K_i parameters of the rudder controller that result in the desired movement of the sailboat according to each application.

The proposed solution consists of finding the best PI controller for a certain application through simulations. Field experiments show that the variables with the highest influence on the movement of the sailboat are the initial orientation (θ) and the wind direction (α_{aw}). Therefore, the following method takes into account changes only in those two variables. Initially, it is necessary to identify the restrictions associated with the desired mission in which the sailboat will be applied. Then, the initial conditions of the simulation and the parameters of the controller are discretized. The initial heading is divided into eight cases ($\mathbf{0}^\circ$, $\pm 45^\circ$, $\pm 90^\circ$, $\pm 135^\circ$, 180°) and the wind direction is separated into four cases (45°, 90°, 135°, 180°). Each pair of initial conditions represents a maneuver in which a pair of satisfactory control parameters must be found according to the established restrictions. In each case, simulation tests are performed, varying the values of K_p and K_i and the best response for the maneuvering is stored in a table. At the end of the process, a control table is assembled, containing the best pairs found for each of the 32 established maneuvers. A static PI controller is found for the same application, in order to compare its performance with that of the GS-PI. The strategy used to find this controller (static PI) is to identify the worst case for the application, that is, the maneuvering with the worst result. The parameters of the static PI controller are then defined as a pair (K_p, K_i) that produces the best result for this maneuvering and are found by trial and error, a common approach in sailboat control.

The mission of water quality monitoring in rivers is used to test the described method. In this application, the available navigation area is narrow, which implies a smaller area for the execution of the maneuverings. In this case, the controller must keep the sailboat in a safe zone, which may be previously known by a higher level agent looking a map for example or obtained in real time using cameras and/or 3-D sensors. In this situation, a control restriction is to ensure that the sailboat is always inside the safe area. It is then necessary to check the available lateral distance (d_l) for navigation when starting a new maneuvering. The proposed solution to this problem is to use a lateral distance restriction, that is, to find the pair (K_p, K_i) that results in the shorter d_l.

Algorithm 1 presents a possible implementation of the above idea. The loops in lines 3 and 5 generate the initial conditions of the maneuvering, which are represented in the algorithm by variables θ and α_{aw}. The loops in lines 7 and 8 vary the parameters K_p and K_i for each wind situation and initial sailboat heading. The pair (K_p, K_i) is then simulated using the *sim* function (line 9). This function modifies the initial conditions of wind direction and the heading for simulation. The *sim* function returns -1 if the sailboat does not reach the target point in an established simulation time. Lines 10–14 store the parameters (K_p, K_i) that resulted in the shortest lateral distance in position (i_s, i_w) of matrix N. These steps are repeated for each wind direction and sailboat heading (lines 3 and 5), so that after the algorithm execution the N matrix will contain the best (K_p, K_i) found for each tested maneuver.

Algorithm 1. Finding the GS-PI controller.

Result: Matrix N with the best found PI parameters.
begin
  $\theta \leftarrow 45$;
  for i_w *from* 1 *to* 4 **do**
    $\alpha_{aw} \leftarrow 45 + (i_w - 1) \times 45$;
    for i_s *from* 1 *to* 8 **do**
      $\theta \leftarrow \theta - 45$; $d_{min} \leftarrow 0$;
      for k_p *from* 1 *with step* 0.5 *to* 10.5 **do**
        for k_i *from* 1 *to* 20 **do**
          $d_l \leftarrow \text{sim}(\theta, \alpha_{aw}, K_p, K_i)$;
          if $d_l \neq -1$ **then**
            if $d_l < d_{min}$ **then**
              $d_{min} \leftarrow d_l$;
              $N_{i_s,i_w} \leftarrow (K_p, K_i)$;
            end
          end
        end
      end
      if $\theta = -180$ **then**
        $\theta \leftarrow 180$;
      end
    end
  end
end
Output: N

Short-term path planning in upwind situations

In this work, the wind direction can be determined in relation to the sailboat and the target direction using a wind vane sensor. The sailboat direction can simply be determined by way of a compass or a more precise sensor in case of a high-precision mission. Moving the sailboat toward a target position is a question of choosing the best strategy if this target position is not directly against the wind. If the desired position is directly against wind, new waypoints have to be calculated (and optimized) coming up with a zigzag path for the tacking. This new path will be composed of a series of alternated waypoints, to one side and to another side of the straight line defined by the actual sailboat position to the target position, which composes the tacking maneuvering. The amount of waypoints to be used and associated zigzag angles are determined as a function of the surrounding conditions (available area for maneuvering) and on the mission requirements, providing the sailboat navigation to the goal as fast as desired.

This upwind path planning method has to find points in-between and laterally to the straight line linking the actual sailboat position and the desired position, in order for the sailboat to navigate to the target in situations that it is directly against the wind. Our proposal for solving this situation is to create a path planning method that can be used to determine alternated waypoints, in such a way that all of them can be reachable, from one to the next one. This condition is fulfilled if the paths from each of them to the next are outside the dead zone, which implies that they are reachable in straight line. This can be done by choosing appropriate tacking distance d_t and orientation θ_t parameters as shown in Figure 3. In the figure, notice that the parameters that define the equation of line A can be determined using the actual sailboat position $P_0 = (x_0, y_0)$ and the target position $P_d = (x_d, y_d)$, as given by equation (22)

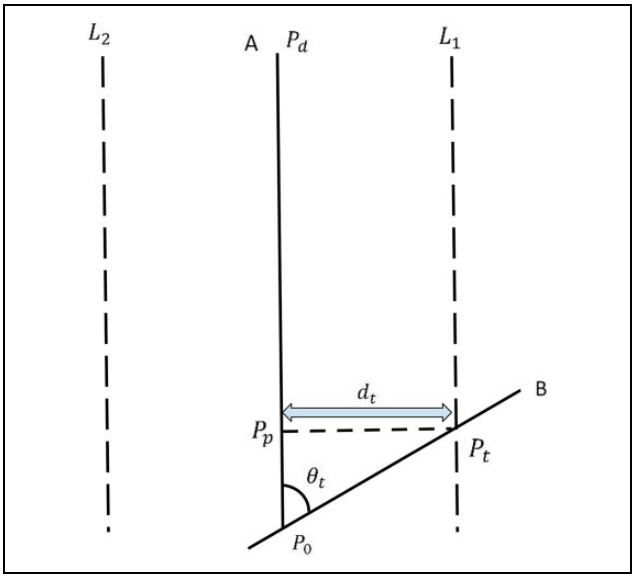

Figure 3. Path planning method illustration, where points P_0 (actual sailboat position) and P_d (target position) are used joint with d_t and θ_t parameters, which are the tacking distance and orientation, respectively, in order to determine the new waypoints for the tacking.

$$a_A = \frac{y_d - y_0}{x_d - x_0}; \qquad b_A = y_0 - a_A x_0 \tag{22}$$

From these, the parameters of the line B, obeying an angle of θ_t with A, can be determined by way of equation (23). This angle θ_t, which can be empirically chosen, keeps the sailboat out of the dead zone

$$a_B = \frac{a_A + \tan(\theta_t)}{a_A \tan(\theta_t) - 1}; \qquad b_B = y_0 - a_B x_0 \tag{23}$$

A point P_t can then be determined on the line B, respecting a maximum distance d_t to line A, which is calculated using P_p and P_t. The coordinates x_t and y_t of point P_t are used by equation (24) in order to determine the parameters of lines L_1 and L_2. Notice that these lines are parallel to each other and to line A, and they have the same distance to line A. This provides a laterally limited region around the previous straight path (line A) in the scenario of a narrow (limited) navigation environment

$$b_{L_1} = y_t - a_A x_t; \qquad b_{L_2} = 2b_A - b_{L_1} \tag{24}$$

The number of tacking points is defined by equation (25), that gives the quantity k of steps with size $d_{\mathbf{0},p}$ that are necessary to go from $P_{\mathbf{0}}$ to P_d

$$k = \left\lfloor \frac{d_{0,d}}{d_{0,p}} \right\rfloor \tag{25}$$

The X and Y necessary steps for advancing a certain distance $d_{\mathbf{0},p}$ on the $P_{\mathbf{0}} - P_d$ direction are given by equation (26)

$$\Delta x_{p,0} = x_p - x_0; \qquad \Delta y_{p,0} = y_p - y_0 \tag{26}$$

After these initial calculations, all of the tacking point (x, y) coordinates to follow the path can be found. With the sailboat starting at $P_{\mathbf{0}}$, for $k = 1, 2, \ldots, n$, a new point P_k on the line A can be calculated by advancing with $k\Delta x$ and $k\Delta y$ steps. In even steps, tacking points are given by the projections of the current point P_k in the line L_1. For odd steps, it is just a matter of switching to line L_2. If the target is not directly against the wind but the sailboat is still in the dead zone, that is $e \neq \mathbf{0}$ (equation (27)), then the distance between odd and even tacking points is changed in order to take the wind angle into account. This allows the sailboat to keep the same tacking angle according to the wind, throughout the maneuvering, increasing the robustness of our method

$$e = |\alpha_{tw} - \theta_t| \tag{27}$$

Notice that by changing the parameters θ_t and d_t, the behavior of this approach will also change resulting in a different tacking strategy. That is to say that different values of them imply in different tacking points. A possible implementation of our method is given in Algorithm 2.

Algorithm 2. Short-term path panning method.

Input: $P_0, P_d, \theta_t, d_t, \alpha_{aw}, \theta_d, \theta$
Result: Matrix N containing the tacking points
begin
  if $|\alpha_{aw}| < 30$ *and* $|\theta_d - \theta| < 5$ **then**
    $a_A \leftarrow \frac{y_d - y_0}{x_d - x_0}$; $b_A \leftarrow y_0 - a_A x_0$;
    if $\theta - \theta_d < 0$ **then**
      $a_B \leftarrow \frac{-a_A + \tan(\theta_t)}{-a_A \tan(\theta_t) - 1}$;
    else
      $a_B \leftarrow \frac{-a_A - \tan(\theta_t)}{a_A \tan(\theta_t) - 1}$;
    end
    $b_B \leftarrow y_0 - a_B x_0$;
    $x_p \leftarrow x_d$; $y_p \leftarrow y_d$;
    repeat
      $x_p \leftarrow \frac{x_p + x_0}{2}$; $y_p \leftarrow \frac{y_p + y_0}{2}$;
      $P_t \leftarrow Proj_{p,B}$;
      $d \leftarrow d_{p,t}$;
    until $d \leq d_t$;
    $b_{L_1} \leftarrow y_t - a_A x_t$; $b_{L_2} \leftarrow 2b_A - b_{L_1}$;
    $\Delta x \leftarrow x_p - x_0$; $\Delta y \leftarrow y_p - y_0$;
    $P_{0_{L_1}} \leftarrow Proj_{m,L_1}$; $P_{0_{L_2}} \leftarrow Proj_{m,L_2}$;
    $k \leftarrow \left\lfloor \frac{d_{0,d}}{d_{0,p}} \right\rfloor$;
    for $z \leftarrow 1$ *to* $k - 1$ **do**
      $\Delta x_t \leftarrow z\Delta x$; $\Delta y_t \leftarrow z\Delta y$;
      if $mod(z, 2) = 0$ **then**
        $N_{z,1} \leftarrow x_{0_{L_1}} + \Delta x_t$;
        $\mathrm{N}_{z,2} \leftarrow y_{0_{L_1}} + \Delta y_t$;
      else
        $e \leftarrow |\alpha_{tw} - \theta_t|$;
        $\Delta x_t \leftarrow \Delta x_t - e \times \frac{\Delta_x}{\lambda}$;
        $\Delta y_t \leftarrow \Delta y_t - e \times \frac{\Delta_y}{\lambda}$;
        $\mathrm{N}_{z,1} \leftarrow x_{0_{L_2}} + \Delta x_t$;
        $\mathrm{N}_{z,2} \leftarrow y_{0_{L_2}} + \Delta y_t$;
      end
    end
  end
end
Output: N

Optimization of the tacking approach

Parameters θ_t and d_t should be empirically determined, in order for this proposed tacking approach work properly. Variations in these parameters imply faster or slower trajectories, wider or narrower maneuvering area, or the creation of undesired tacking points as the situation shown in the left-hand graph of Figure 8. The number of tacks is the parameter that most influences the time to target of a beating maneuvering.[34] The problem is that for each maneuvering around the tacking points, the sailboat drastically reduces its velocity as its has to change its direction passing through the dead zone. If both the directions of the water current and wind do not vary, the best strategy is to perform minimum amount of tacking, which would be one. This is not always feasible due to environment restrictions nor possible due to variations in the directions of the wind or water current. We also notice here that, in practice, experienced sailors try not to deviate so long from the straight line to the target due to possible changes in wind direction (and water current) that would bring the sailboat to a dangerous situation or at least far from the target position. In fact, this strategy has sometimes made the difference for a team to be the champions of a regatta.

For the optimization of the tacking, we initially try to describe the function to be optimized through a series of initial experiments. This is done in a straightforward way by varying the values of the parameters (θ_t, d_t), so a brute force method is used to encounter optimal values of θ_t and d_t regarding time to target. The parameter θ_t varies from 20 to 80 and the d_t varies from 10% to 90% of the initial

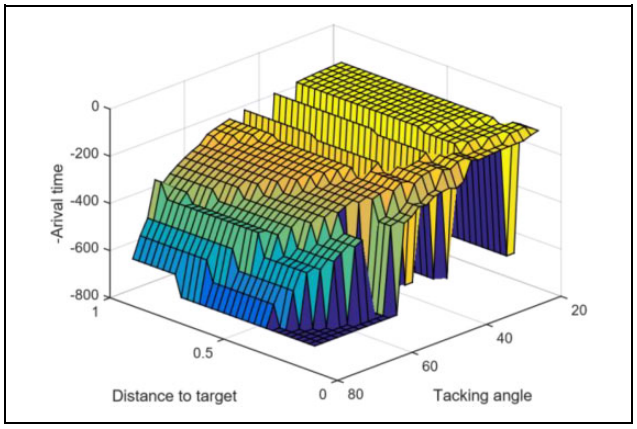

Figure 4. Brute force method graph obtained in simulation with varying parameters d_t and θ_t for scenario 2.

distance to target. The times to target for one of these experiments are shown in Figure 4, notice that several local minima appears in the experiment result. Therefore, an approximation to the global minimum can be found by way of using a GA approach. The main reason for using GA is that this problem relates to a huge search space given several parameters, accounting for the several variations that may happen (heading and initial speed of the sailboat, wind direction and wind speed). Varying each one of these may result in a different time to target. Notice that this optimization should be applied every time that a waypoint is set, what happens in general at the beginning of each maneuvering. So the system just apply this proposed technique checking its status and using it as the initial condition at the simulator. The problem is divided in several scenarios and the performance and validity of the technique are analyzed for each specific scenario. After that step, this procedure can be embarked in the sailboat for practical use.

Algorithm 3 implements the idea above, using a traditional GA approach for the short-term path planning optimization. Binary strings represent individuals and they contain values for θ_t and d_t. First, an initial population containing n individuals is randomly generated. The simulator calculates the fitness value for each individual of the population by simulating their θ_t and d_t. The fitness of an individual is defined according to its time to target (time taken to get from start to desired position). Individuals with shorter time to target are given higher fitness values in order to minimize time to target. The roulette is then used for performing the crossover step over a selected part of the population. The probability of being selected is higher for individuals with greater fitness values, so the crossover can be effectively performed. Each portion that represents the binary θ_t and d_t allows to select the respective points, performing an exchanging of their tails. For each individual, every bit is tested with the mutation probability that is present in each bit. The best individuals of this population

Algorithm 3. Optimization algorithm for short-term path planning.

Input: parameters of the genetic algorithm
Result: θ_t and d_t that are optimal
begin
  initialization of parameters of genetic algorithm;
  random generation of initial population;
  for $i \leftarrow 1$ *to* N_{ger} **do**
    foreach *individual* **do**
      find the time to target using individual's θ_t and d_t;
    end
    compute fitness;
    apply roulette selection;
    apply crossover;
    apply mutation;
    store best individual;
    generate new population;
  end
end
Output: best individual (θ_t, d_t)

are selected to compose the new population, together with newly reproduced individuals. This process repeats for a predefined number of generations. At the end, the individual with the highest fitness in the population is selected to generate the tacking points. Nonetheless, the above variable gains K_p and K_i as defined by the GS-PI procedure can also be used here completing, thus, the necessary general control behavior of the sailboat.

Experiments and results

The proposed navigation technique, as a whole, is analyzed through a series of experiments, which were planned for verifying and validating both methods devised here. Experimental setups and tests are described next for the GS-PI control approach and for the short-term path planing, respectively.

Experiments with the control system

We performed a series of simulated experiments to test the performance and robustness of the GS-PI when compared with a static PI approach and with a simpler P controller ($K_p = 1, K_i = \mathbf{0}$). The navigation mission occurs in narrow environments where there are restrictions of the areas for performing the maneuvering. The initial state vector for the experiments is $[\mathbf{0}; \mathbf{0}; \mathbf{0}; \theta; u; \mathbf{0}; \mathbf{0}; \mathbf{0}]$, where θ is modified by the method and $u = 2$ m/s. The wind velocity is set as constant at 8 m/s .

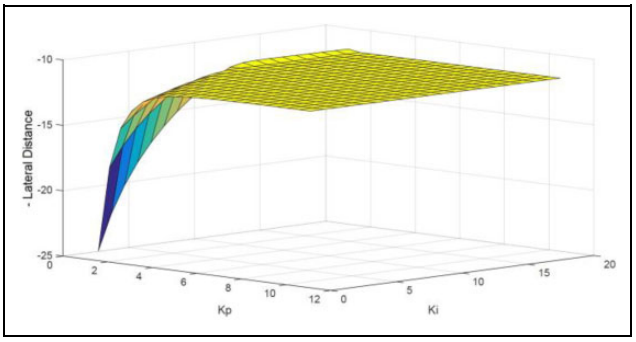

Figure 5. Graphic representation of the results obtained using the exhaustive search method.

Finding the static PI controller

For the application of navigation in narrow environments, the worst scenario happens when the wind direction and initial sailboat orientation result in a route with maximum lateral distance. Simulation tests have shown that this is the case when the wind direction is 180° and the initial orientation of the sailboat is 135°. Figure 5 shows the simulation result using the exhaustive search method (brute force). The values of K_p and K_i vary and the maximum lateral distance is obtained for each maneuvering. This experiment shows that for the worst scenario, the minimum lateral distance occurs when $(K_p, K_i) = (1, 14)$.

Finding the GS-PI controller

The experiment to find the GS-PI controller is similar to the static PI, but now the same experiment is applied for several wind direction and initial sailboat orientations. The problem was then divided into 32 cases: 4 wind directions $(45°, 90°, 135°, 180°)$ and 8 initial orientations ($\mathbf{0}°$, $-45°$, $-90°$, $-135°$, $180°$, $135°$, $90°$, $45°$). The Table 3 shows the pairs (K_p, K_i) obtained with the Algorithm 1.

Discussion

Figures 6 and 7 show the results of the experiments using three control strategies: P, static PI, and GS-PI. The test shown in Figure 6 was performed with wind coming from 180 ° and initial heading of 45°. In this case, all controllers were able to keep the sailboat within the safe zone, but the GS-PI controller reduced the lateral distance by 46.21% when compared to the static PI controller.

Figure 7 shows the result of another experiment carried out, where the wind comes from 90° and the initial orientation of the sailboat is $-90°$. In this experiment, the controller P was unable to keep the sailboat in the safe zone. Static PI and GS-PI controllers met this requirement, with the GS-PI controller reducing the lateral distance by 27.38% when compared to the static PI controller. In general, the experiments show that the GS-PI controller is the most suitable, among the analyzed controllers, for applications in narrow environments. A size reduction of the GS-PI controller table is also a possibility, since there are some cases in which the initial orientations ($-45°$, $-90°$, $-135°$, and 45°) have produced similar control parameters, indicating that the search space could be reduced.

Table 3. Pairs (K_p, K_i) of GS-PI controller.

$\theta \backslash \alpha_{tw}$	45°	90°	135°	180°
0°	(9, 18)	(1, 13)	(10.5, 20)	(10.5, 20)
−45°	(10.5, 20)	(10.5, 20)	(10.5, 20)	(10.5, 20)
−90°	(1.5, 20)	(1.5, 20)	(1.5, 20)	(1.5, 20)
−135°	(1, 15)	(1, 15)	(1, 16)	(1, 14)
180°	(10.5, 2)	(10.5, 2)	(1, 1)	(10.5, 2)
135°	(1, 14)	(1, 13)	(2.5, 1)	(1, 14)
90°	(1.5, 20)	(10.5, 13)	(3.5, 17)	(1.5, 20)
45°	(10.5, 20)	(10.5, 20)	(10.5, 20)	(10.5, 20)

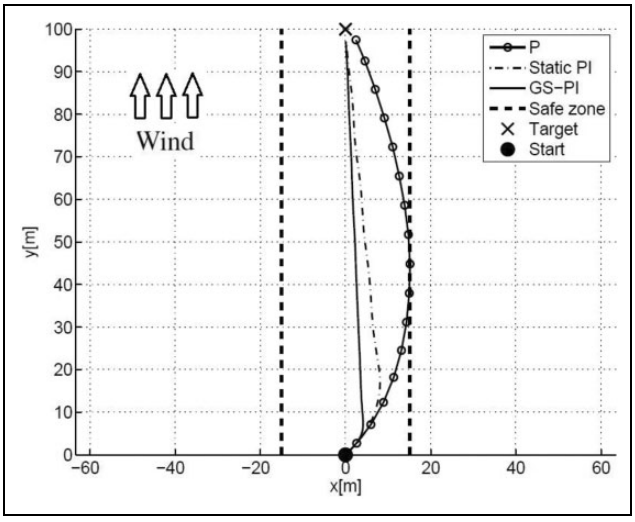

Figure 6. Sailboat path during first control experiment. Initial heading = 45°. Wind direction = 180°.

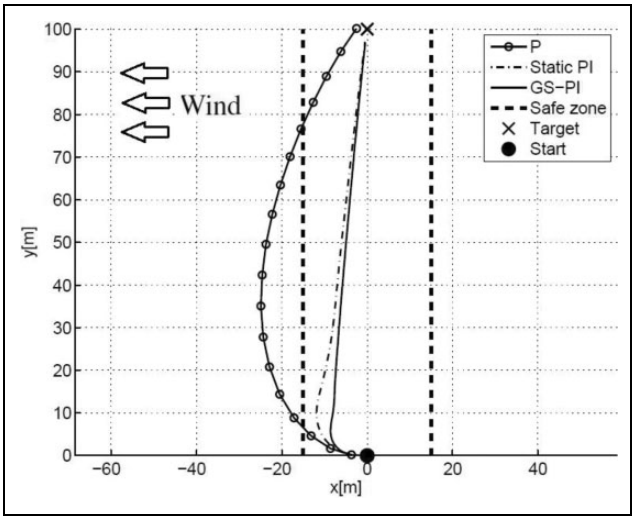

Figure 7. Sailboat path during second control experiment. Initial heading = −90°. Wind direction = 90°.

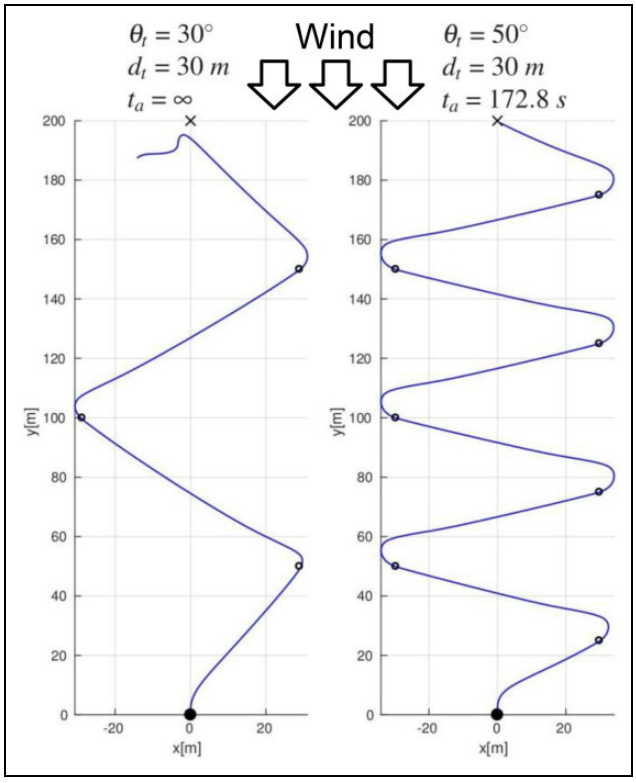

Figure 8. Trajectory against wind from simulated experiment with constant d_t and varying θ_t. The direction of the wind is $\mathbf{0}^\circ$. The black dot is the initial position, black cross is the target position, and the generated tacking points are the black circles. The sailboat trajectory is in blue. The parameters used for the left-hand graph are set as $[\theta_t, d_t, t_a] = [30, 30, \infty]$ and for the right-hand graph are set as $[50, 30, 172.8]$.

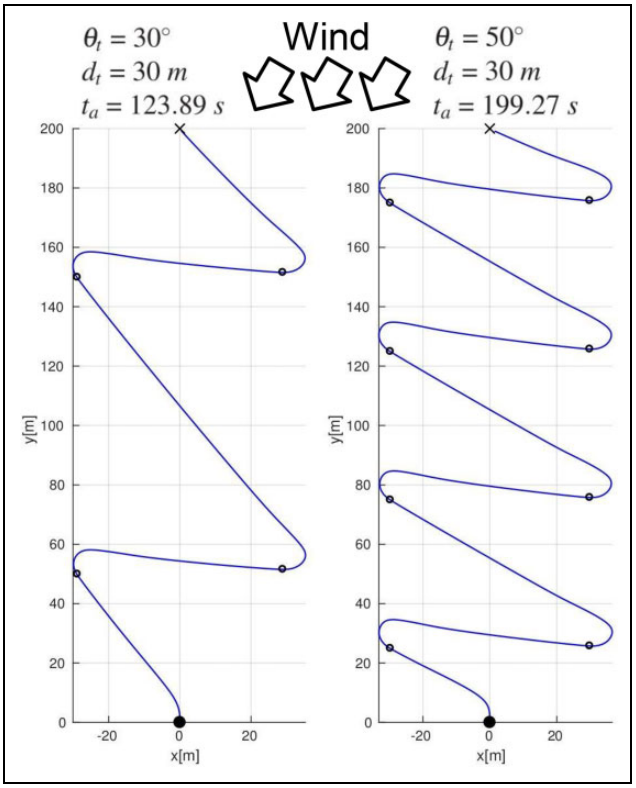

Figure 9. Trajectory against wind with 25° wind angle. The black dot is the initial position, black cross is the target position, and the generated tacking points are the black circles. The sailboat trajectory is in blue. The parameters used for the left-hand graph are set as $[\theta_t, d_t, t_a] = [30, 30, 123.89]$ and for the right-hand graph are set as $[50, 30, 199.27]$.

Experiments with the short-term path planning

For testing the path planning for upwind situations, without loosing generality, lets set the boat starting point at $(\mathbf{0}, \mathbf{0})$ with heading of $\theta = \mathbf{0}^\circ$ and the initial sailboat velocity in x of 2 m/s. In the first experiment, the wind comes from the North $((\mathbf{0}, 200) - (\mathbf{0}, \mathbf{0})$ direction) which is 0°. The target position is set to be directly against the wind direction, at $(\mathbf{0}, 200)$. Figure 8 shows the resulting route for this experiment, in which the parameter d_t is set to 30 m and the parameter θ_t varies. This represents the situation in which there is a limitation in the movement of the sailboat. This example also shows the increase of the tacking points when the value of θ_t goes up. Figure 9 shows an experiment with different initial wind direction set to 25° (still in the dead zone). Notice that in this case, non-symmetrical tacking points were generated in order to allow the sailboat to reach the target. The results show that, under different conditions of wind, the method produces different tacking points in order to reach the same target. In these cases, the lowest time to target paths are those with fewer tacking points. Figure 10 shows a 30 min-simulated experiment with the fully operational system (low-level control and tacking). In this experiment, we set 3 waypoints which the sailboat need to reach, simulating a water monitoring mission in a lake.

Optimizing the path

In the following, we use the GA with the tacking method described above to find near optimal tacking parameters that improve time to target. The performance of this GA approach is compared against a brute force method, which consists of setting the tacking parameters by trial and error similar to what was presented in Figures 8 and 9. To validate this optimization method, we separated the problem in scenarios. The best parameters, which in this scenario are the parameters that result in the minimum time to target, are determined by both methods for each scenario shown in Table 4.

For the brute force method, the parameters θ_t and d_t are discretized and simulated for each scenario. Then, the parameters that have generated the lowest time to target are stored. The GA is executed 10 times for each scenario. The averages of the running time for the experiment and the time to target during the experiment are computed and shown in Table 5. The performance of the tacking points found with GA is quite similar to the points found by brute

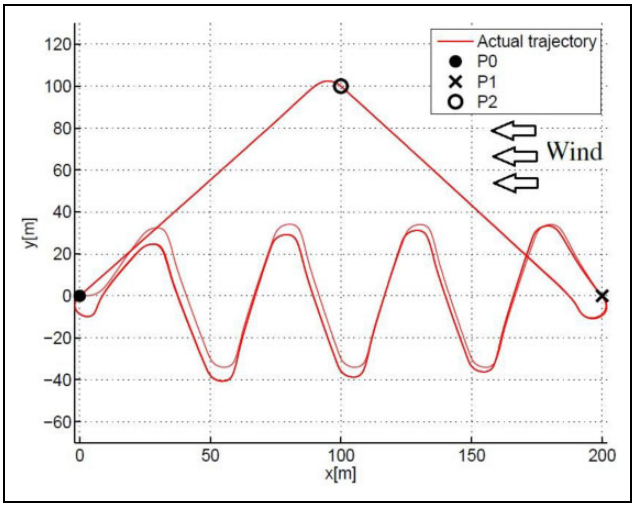

Figure 10. Simulated experiment with 30 min of duration, in which the low-level control and tacking are fully operating in the navigation system.

Table 4. Wind scenarios.

Scenario	Wind direction	Wind velocity
1	60°	10 m/s
2	90°	15 m/s
3	60°	15 m/s

Table 5. Results obtained by both approaches.

Scenario	Method	Time to target	Simulation time
1	Brute force	102.4 s	2364.3 s
	GA	97.72 s	176.48 s
2	Brute force	105.2 s	2172.42 s
	GA	114.44 s	216.28 s
3	Brute force	82 s	2559.96 s
	GA	80.16 s	196.32 s

force. Furthermore, the use of GA substantially reduces the simulation time in comparison to brute force. The GA also found better tacking points for scenarios 1 and 3. This happened because the GA has better precision and can reach points in the search area that are not reached by the brute force approach due to the error introduced by the discretization (the precision used is not enough to reach some points). The GA has not obtained the best result in the second scenario as shown in Figure 11, which indicates that for this case the GA got stuck in a local minimum. Even in this scenario, the GA solution is only 8.87% worse than the best value found by brute force.

Conclusion

A navigation system consisting of both a short-term path planning and a GS-PI-based low-level heading controller is

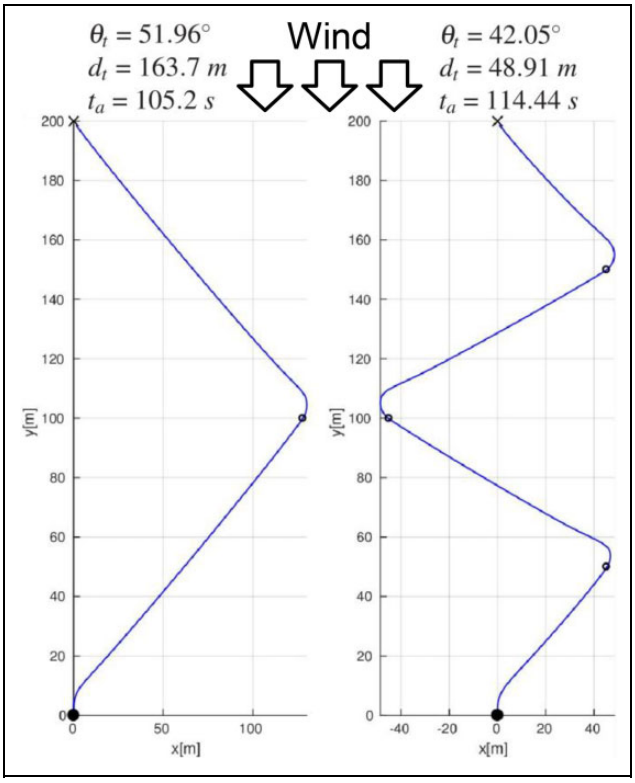

Figure 11. Comparison of brute force with our approach in scenario 2 using the best tacking points found by both approaches. Values for the parameters $[\theta_t, d_t, t_a]$ are set as $[51.96, 163.7, 105.2]$ for the left-hand trajectory and as $[42.05, 48.91, 114.44]$ for the right-hand one.

proposed in this work that allows a sailboat robot to reach desired positions autonomously. For the path planning, we apply a deterministic approach for immediate (short-term) path redefinition, and a further optimization by GA, in order to get the best tacking points according to the space available for maneuvering and appropriate navigation angle. Experimental results verify and validate our approach. In all of the experiments with the optimized method, the sailboat was able to reach the final target point. The method is versatile, if we consider that the two parameters (θ_t and d_t) allow to leave the task of defining the sailboat high-level behavior to a higher hierarchical agent, for example, to change the tacking angle in order to get more roll stability during the sailing. Our optimization procedure has produced similar or better results when compared to a brute force approach, however the optimized path planning is about 10 times faster than the brute force.

As a second contribution, we propose to apply a GS strategy to enhance the low-level control, coming up with a new PI control approach not used previously in robotic sailboats. Several pairs of parameters (K_p, K_i) were found for 32 specified wind direction and initial heading scenarios. The presented results verify the efficacy of the controller when compared to a conventional PI control strategy commonly used by sailboat projects in the literature, where

only one single pair (K_p, K_i) is used during the whole mission. In the experiments performed, the GS-PI controller is able to keep the sailboat within the safe area also considerably reducing the lateral distance.

The data obtained in the simulation experiments confirm the initial hypothesis that the use of a GS-PI controller can improve the sailboat performance, mainly during area constrained missions, when compared to PI controllers that use static parameters. The strategy to find a GS-PI controller introduced in this article can be automated by way of using the mathematical model of a sailboat. The main drawback for the design of a GS-PI controller is to find suitable models that faithfully represent the movement of the sailboat. The implementation of the method developed in the N-Boat II and field trials to investigate the impact that random variables such as water currents and waves have on the movement of the sailboat are possibilities of future work.

Future work, that has already been started, consists on implementing and testing in practice the versions of the methods proposed for this navigation system, in a larger sailboat that has finished its construction. Further tests will be performed in real situations of sailing. This further implementation will allow the reduction of computational complexity that is a necessity for the real-time usage of the developed tools in embedded processors of this larger sailboat robot.

Acknowledgements

We thank Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil, for the grants of Davi Santos under finance code 001, and National Sponsoring Agency for Research (CNPq) for the grants of Luiz Gonçalves.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This study was financed by Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil, under finance code 001 and research project number 88887.123914/2015-00.

ORCID iD

Luiz Marcos Garcia Goncalves https://orcid.org/0000-0002-7735-5630

References

1. AS OS. Sailbuoy—unmanned surface vessel, http://sailbuoy.no/ (2018, accessed 14 September 2018).
2. Cruz NA and Alves JC. Auto-heading controller for an autonomous sailboat. In: *OCEANS 2010 IEEE—Sydney*, Sydney, 24–27 May 2010, pp. 1–6. DOI: 10.1109/OCEANSSYD.2010.5603882.
3. Erckens H, Beusser GA, Pradalier C, et al. Avalon. *IEEE Robotics & Automation Magazine* 2010; 17(1): 45–54.
4. Tranzatto M, Grammatico S, Liniger A, et al. The debut of Aeolus, the autonomous model sailboat of ETH Zurich. In: *IEEE oceans conference*, Genova, 18–21 May 2015.
5. Melin J. Modeling, control and state-estimation for an autonomous sailboat. 2015; 15023: 42.
6. Williamsz B, Kane J, Hartnett RJ, and Swaszek PF. Senior project design of a two meter autonomous sailboat. In: *Proceedings of the 2014 International Technical Meeting of The Institute of Navigation*, San Diego, California, January 2014, pp. 594–600.
7. Kumar P and Ippolito C. Modeling and trajectory control of an autonomous sailboat for path planning. In: *AIAA Infotech@Aerospace conference—modeling and trajectory control of an autonomous sailboat for path planning*, Seatle, Washington, 6–9 April 2009. DOI: 10.2514/6.2009-1965.
8. Cruz NA and Alves JC. Navigation performance of an autonomous sailing robot. In: *2014 Oceans—St. John's*, St. John's, 14–19 September 2014, pp. 1–7. DOI: 10.1109/OCEANS.2014.7003227.
9. Dalang RC, Dumas F, Sardy S, et al. Stochastic optimization of sailing trajectories in an upwind regatta. *Journal of the Operational Research Society* 2015; 66(5): 807–821. DOI: 10.1057/jors.2014.40.
10. Płtrs C, Romero-Ramirez MA, and Plumet F. A potential field approach for reactive navigation of autonomous sailboats. *Robotics and Autonomous Systems* 2012; 60(12): 1520–1527. DOI: 10.1016/j.robot.2012.08.004.
11. Zhu J, Wang J, Zheng T, et al. Straight path following of unmanned surface vehicle under flow disturbance. In: *OCEANS 2016—Shanghai*, IEEE, Shanghai, 10–13 April 2016, pp. 1–7.
12. Sohn SI, Oh JH, Lee YS, et al. Design of a fuel-cell-powered Catamaran-type unmanned surface vehicle. *IEEE Journal of Oceanic Engineering* 2015; 40(2): 388–396.
13. Kragelund S, Dobrokhodov V, Monarrez A, et al. Adaptive speed control for autonomous surface vessels. In: *Oceans—San Diego, 2013*, IEEE, San Diego, 23–27 September 2013, pp. 1–10.
14. Larrazabal JM and Peñas MS. Intelligent rudder control of an unmanned surface vessel. *Expert Systems with Applications* 2016; 55: 106–117.
15. Borge RS. Application of the unmanned offshore sensing Sailbuoy for validation of ocean model simulations and remote sensing data in the North Atlantic. PhD Thesis, *University of Bergen*, Bergen, Norway, 2015.
16. Hole LR, Fer I, and Peddie D. Directional wave measurements using an autonomous vessel. *Ocean Dynamics* 2016; 66(9): 1087–1098. DOI:10.1007/s10236-016-0969-4.
17. de Canarias G. Plataforma oceánica de canarias, http://www.plocan.eu/index.php/en/ (2018, accessed 14 September 2018).
18. Briere Y. Iboat: an autonomous robot for long-term offshore operation. In: *MELECON 2008—The 14th IEEE*

Mediterranean electrotechnical conference, Ajaccio, France, 5–7 May 2008, pp. 323–329. DOI: 10.1109/MELCON.2008.4618455.

19. Stelzer R and Jafarmadar K. The robotic sailing boat ASV roboat as a maritime research platform. In: *Proceedings of 22nd international HISWA symposium*, Amsterdan, 12–13 November 2012.
20. Gomes L, Costa A, Fernandes D, et al. *Improving instrumentation support and control strategies for autonomous sailboats in a regatta contest*. Cham: Springer International Publishing, 2017, pp. 45–56. ISBN 978-3-319-45453-5.
21. Furrer F and Gohl PJ. Control of an autonomous sailing Catamaran on the concept of hydrofoils. Semester thesis, ETH Zürich, Zürich, Switzerland, 2010.
22. Le Bars F and Jaulin L. *An Experimental validation of a robust controller with the VAIMOS autonomous sailboat*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 73–84. ISBN 978-3-642-33084-1. DOI: 10.1007/978-3-642-33084-1_7.
23. Saoud H, Hua MD, Plumet F, et al. Routing and course control of an autonomous sailboat. In: *2015 European conference on mobile robots (ECMR)*, Lincoln, UK, 2–4 September 2015, pp. 1–6. DOI: 10.1109/ECMR.2015.7324218.
24. Plumet F, Saoud H, and Hua MD. Line following for an autonomous sailboat using potential fields method. In: *2013 MTS/IEEE OCEANS—Bergen*, Bergen, Norway, 10–13 June 2013, pp. 1–6. DOI: 10.1109/OCEANS-Bergen.2013.6607961.
25. Aouf N and Boulet B. Linear fractional transformation gain-scheduling flight control preserving robust performance. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 2012; 226(7): 763–773. DOI: 10.1177/0954410011415767.
26. Sadeghzadeh I, Mehta A, and Zhang Y. Fault-tolerant control of quadrotor helicopter using gain-scheduled PID and model reference adaptive control. *Journal of Unmanned System Technology* 2016; 3(3): 108–118.
27. Silvestre C, Cunha R, Paulino N, et al. A bottom-following preview controller for autonomous underwater vehicles. *IEEE Transactions on Control Systems Technology* 2009; 17(2): 257–266. DOI: 10.1109/TCST.2008.922560.
28. Silvestre C and Pascoal A. Depth control of the INFANTE AUV using gain-scheduled reduced order output feedback. *Control Engineering Practice* 2007; 15(7): 883–895. DOI: 10.1016/j.conengprac.2006.05.005. Special Issue on Award Winning Applications 2005 IFAC World Congress.
29. Yang Y and Yan Y. Neural network gain-scheduling sliding mode control for three-dimensional trajectory tracking of robotic airships. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 2015; 229(6): 529–540. DOI: 10.1177/0959651815570353.
30. Stelzer R and Prll T. Autonomous sailboat navigation for short course racing. *Robotics and Autonomous Systems* 2008; 56(7): 604–614. DOI: 10.1016/j.robot.2007.10.004.
31. Philpott A and Mason A. Optimising yacht routes under uncertainty. In: *The 15th Chesapeake sailing yacht symposium*, Annapolis, Maryland, USA, 26–27 January 2001.
32. Ferguson DS and Elinas P. A Markov decision process model for strategic decision making in sailboat racing. In: *Canadian conference on artificial intelligence*, St. John's, 25–27 May 2011, pp. 110–121. Springer.
33. Xiao L and Jouffroy J. Modeling and nonlinear heading control of sailing yachts. *IEEE Journal of Oceanic Engineering* 2014; 39(2): 256–268.
34. Stelzer R. *Autonomous sailboat navigation: novel algorithms and experimental demonstration*. PhD Thesis, Centre for Computational Intelligence, De Montfort University, Leicester, UK, 2012.