Simulated Control and Path Planning of an Autonomous Sailboat

Andrew Fearing, Neelay Junnarkar, Hamza Kamran Khawaja

Abstract—In this project we develop a planning and control method for a sailboat to maneuver autonomously, in the presence of obstacles, to a high level target described in x, y coordinates. We test our method in the stda−sailboat−simulator against various wind directions and with and without presence of waves.

I. Introduction

Wind-propelled sailing is a complex task. The sailboat is entirely dependent on the wind and water for propulsion and steering. In addition, boats are subject to environmental disturbances such as wind gusts, water currents, and waves. Because of this, autonomous sailing presents an interesting problem in robotics.

Sailboats have the potential to go on long-term missions at sea. They need actuators only for the rudder and sail, and so they can be solar powered. Possible uses include plastic waste collection, coastal surveillance, and long-term data collection.

In this paper, we present an application of an optimization-based path planner to successfully navigate the simulated sailboat to a target position in the presence of obstacles and under different environmental conditions. In particular, we demonstrate the sailboat succesfully traversing a narrow channel. The sailboat dimensions, kinematics, and dynamics model is given in [1]. Our approach decouples high-level planning from lower-level heading control, leveraging heading controllers found in papers such as [1].

II. Related Work

Interest in autonomous sailboats has grown over the past decade. There are several different areas of research in pursuit of sailing USV's, including propulsion, control, navigation, and simulation.

Previous research in model-based sailboat control have used PID controllers for heading, such as in [2], [3]. [4] used a nonlinear heading controller using the integrator backstepping method. [5] used a lookup table of PI parameters to suit the control requirements of different environmental conditions.

Efforts in propulsion research [6]–[8] focus on optimizing the sail angle and shape. There are several criteria that can be evaluated, such as time to target, energy consumption, safety, and aerodynamic forces.

There are several methods that have been used for autonomous sailboat path planning. A*, theta*, and D* algorithms have previously been successful [3]. Path planning using potential fields is promising. They offer a way to unify the navigation factors (goal, obstacles, boat characteristics, costs of maneuvers) into a single framework [9]. The typical problems of local minima are not an issue since open water environments are sparse.

name	position	velocity
	x_g	v_x
	y_g	v_y
	z_g	z_x
heading	ψ	p
roll	θ	q
pitch	φ	r
rudder angle	ρ	
sail angle	γ	

TABLE I
The 14 sailboat state variables [1].

One area of that has not found much attention is robustness to environmental effects. Most of the above mentioned literature use some combination of 3- or 4-DoF simulators and practical experiment on a 1 m to 10 m-scale sailboat in fair weather. 3- and 4-DoF simulators usually simplify models by neglecting the roll, pitch, and z-axis motions brought about by waves and wind. Of the open-source USV simulators, the authors were able to find two that support 6-DoF sailboat simulation and model wind, buoyancy, waves, and foil dynamics. [10] demonstrated USVSim, an open-source USV simulator that incorporates Gazebo, the Free-floating Plugin, UWSim, and the Lift-Drag Gazebo Plugin. USVSim includes many features, but it is time-consuming to set up the simulation environment. [1] presented stda−sailboat−simulator , a 6-DoF sailboat simulator written in Python. The simulator uses the RK45 method to solve an ODE over time. Although the simulator does not feature 3D-visualization as Gazebo offers, it requires only a few common Python packages to run.

III. Methods

A. Simulation setup

We decided to use the stda−sailboat−simulator for simplicity, although it required significant refactoring of the code base to be in a useable form.

B. Feedback Loop

Our control design modularizes the system into two stages: a planner, which takes in high level x, y position targets and generates heading trajectories, and a yaw

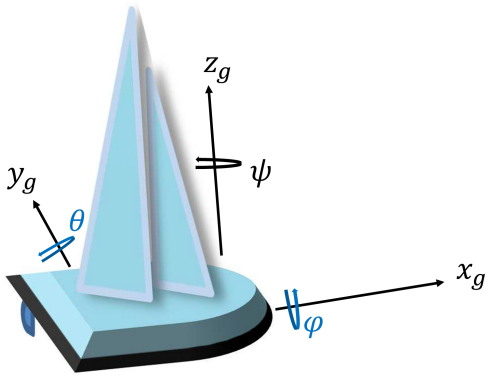

Fig. 1. Sailboat axes [11]

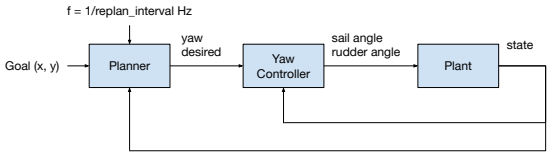

Fig. 2. Block diagram of planner and controller

controller, which generates sail and rudder angles to track the reference heading trajectory output by the planner. Since the planning stage does not need to be run at the same rate as the yaw controller due to the slow speed of the boat, we create an inner feedback loop between the yaw controller and the system dynamics, and an outer feedback loop that incorporates the planner and is run at a lower rate. See Figure 2 for a block diagram.

C. Yaw Controller

The yaw controller is input the full boat state and outputs the rudder and sail angles to set on the boat to track the reference heading trajectory from the planner. Note that in the boat dynamics model, the rudder and sail angles are related to the rudder and sail angle inputs through a first order system that simulates delays, but for brevity we will refer to the outputs of the yaw controller as the rudder and sail angles of the boat.

We implemented the yaw controller as from [1]. This controller leverages the fact that there is an optimal angle for the sail based on wind direction to decouple the rudder and sail angle controls. The sail angle is given by Eq. 1. Note that the full range of this equation is not physically feasible and is clipped to feasible bounds in the model. The rudder angle controller is implemented as a PID controller with gains determined through LQR.

$$\theta_{\text{sail}} = \frac{\sin(\theta_{aw})}{\cos(\theta_{aw}) + 0.4\cos^2(\theta_{aw})} \tag{1}$$

D. Path Planner

The path planner takes in a high level x, y position target and outputs a heading trajectory that if followed by the boat will bring the boat along a path to the target. We designed our planner as an optimization based planner which repeatedly solves constrained optimization problems with receding horizons. Replanning allows handling of deviations from the planned trajectory.

Each optimization problem solves for trajectories of an input u and states q over the time interval t_0, the time at the start of planning, to $t_f = t_0 +$ replanning interval. The constraints are to start at the start state, evolve the state in such a way that it satisfies boat dynamics. We add additional constraints to bound the input. As an extension to handle simple obstacles, we represent obstacles as circles in x, y space. Then, the additional constraint is that the distance between the x, y position of the boat and the center of each obstacle must exceed the radius of each obstacle.

The optimization problem is formulated as in Equation 2, where t_0 is the time at the start of planning, t_f is t_0 plus the planning horizon, u is the input as a function of time from $t = t_0$ to $t = t_f$, $q(t)$ is the state of the boat at time t, q_{t_0} is the state at the start of planning, q_{goal} is the desired state of the boat, u_{max} and u_{min} are bounds on the input u, f_{simple} is a simplified dynamics model, Q and R are tuneable weight matrices, $q_{o_i}(t)$ is the location of obstacle o_i at time t, and r_i is the radius of obstacle i.

$$\begin{aligned}
\min_{u,q} \quad & \int_{t_0}^{t_f} \left((q(t) - q_{goal})^T Q (q(t) - q_{goal}) + u^2(t) R\right) \mathrm{dt} \\
\text{s.t.} \quad & q(t_0) = q_{t_0} \\
& \dot{q}(t) = f_{simple}(q(t), u(t)) \qquad \forall t \\
& \|q(t) - q_{o_i}(t)\|^2 \geq r_i^2 \qquad \forall t, i \\
& u(t) \leq u_{max} \qquad \forall t \\
& u(t) \geq u_{min} \qquad \forall t
\end{aligned} \tag{2}$$

Ideally, f_{simple} would equal the actual dynamics model of the sailboat. However, the dynamics model of the sailboat is large, with 14 state variables, and complicated, with many nonlinear dynamics and involving forces from environmental disturbances not known at the time of planning. Using the real dynamical model would slow down planning, so we instead devise a simplified dynamical model to make the planning efficient [12].

The simplified model we design has three boat states: x, y, ψ where x, y are position and ψ is heading (yaw). The input u is a non-physical input. Setting yaw as the integral of the non-physical input u ensures the yaw trajectory is continuous. Additionally, with this design, constraints on u allow limiting the rate of change of yaw, giving a tuning factor to make the yaw trajectories designed by the planner more feasible on the actual boat model. See Equation 3 for the simplified dynamical model.

$$\dot{q} = f_{simple}(q, u) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v_x \cos(\psi) - v_y \cos(\psi) \\ v_x \sin(\psi) + v_y \cos(\psi) \\ u \end{bmatrix} \tag{3}$$

This dynamical model uses the correct derivatives of x and y, while assuming any slowly and continuously changing heading is feasible. The variables v_x and v_y are the boat velocities in the x and y world frame directions. The speed of the boat is considered to be constant over the course of the planning horizon and equal to the speed at the start of planning.

The solution to the optimization problem gives a trajectory of states $q(t)$, from which the yaw trajectory is extracted and passed to the yaw controller.

IV. Implementation

We used the python stda−sailboat−simulator simulator to model the dynamics of the sailboat, and implemented the path planner and yaw controller as feeding inputs to the simulator. The path planner is implemented using CasADi and the IPOPT solver. To implement Equation 2, we discretized it with a fixed number of steps, discretizing the dynamics constraint as $q_{i+1} = q_i + \delta_t Df$ where δ_t is the discretized step time, and Df is the Jacobian linearization of f_{simple}. The optimization problem is warm started with a q initial trajectory that linearly interpolates q_{start} and q_{goal} in time, and sets the u initial trajectory to all zeros.

The path planner was run with a planning horizon of 50 s and a replanning interval, the time between running the planner, of 50 s. We selected a $Q = 10\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}$ and an $R = 0.1$ to primarily penalize deviation between current x, y and desired x, y.

V. Experiments

The simulation was run in several different cases. Waves, obstacles, and different wind vectors were used. The tests were not exhaustive due to the wide variety of environmental conditions possible at sea and also due to the limitations of the simulator. An experiment was deemed successful if the sailboat got close to the target position.

A. Sailing with wind

To demonstrate that the path planning and control scheme worked, the target position was set to $(100, 40, \pi/4)$ with wind the in the direction of the intended heading, shown in Fig. 3. The wind had a magnitude of 5 m/s.

B. Sailing with crosswind

In this experiment, the wind is changed to point at −45 degrees, roughly at a right angle to the straight line trajectory from start to goal. See Figure 4.

C. Sailing against Wind

In this experiment, the wind points at −135 degrees, and the goal state is $(x, y) = (40, 40)$. This poses an interesting problem since in the real dynamics model there is a null when the boat is heading directly into the wind where the boat has a speed of 0. See Figure 5.

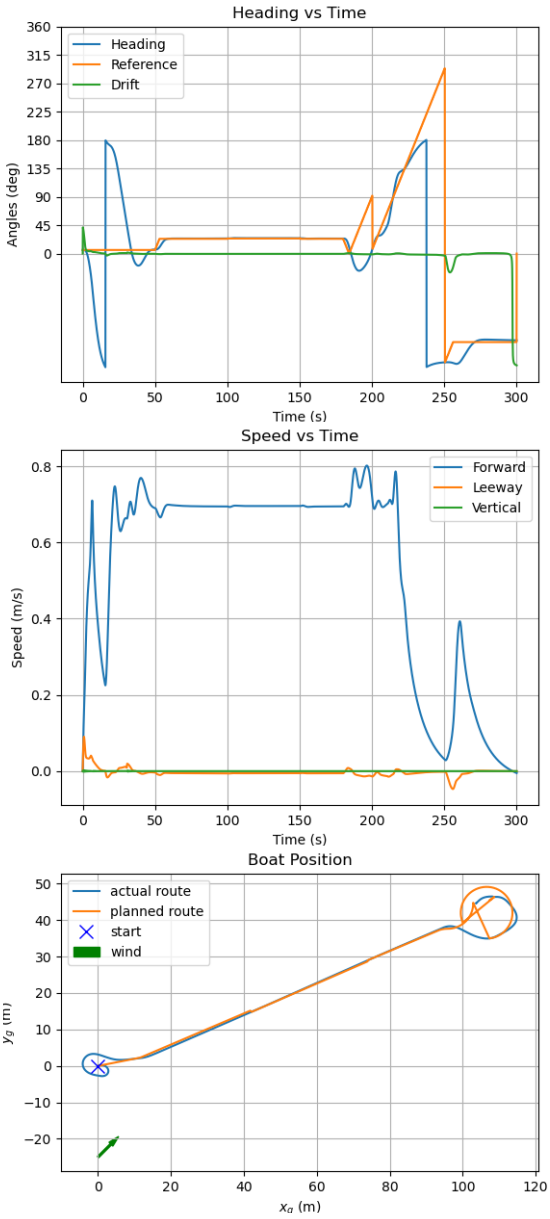

Fig. 3. Sailing with wind: sailing to $(x, y) = (100, 40)$ with wind in the direction of sailing. No waves.

D. Single Obstacle

An obstacle is added at $(x, y) = (60, 30)$ with radius 5. See Figure 6 for the executed plan with and without the obstacle.

E. Channel

Two obstacles are placed: one at $(x, y) = (60, 30)$ with radius 10 and another at $(x, y) = (60, 6)$ with radius 8. These form a channel through which the direct path to $(100, 40)$ must take. See Figure 7. This experiment is run with 5 m/s pointing at 45 degrees, and 0.5 m amplitude waves.

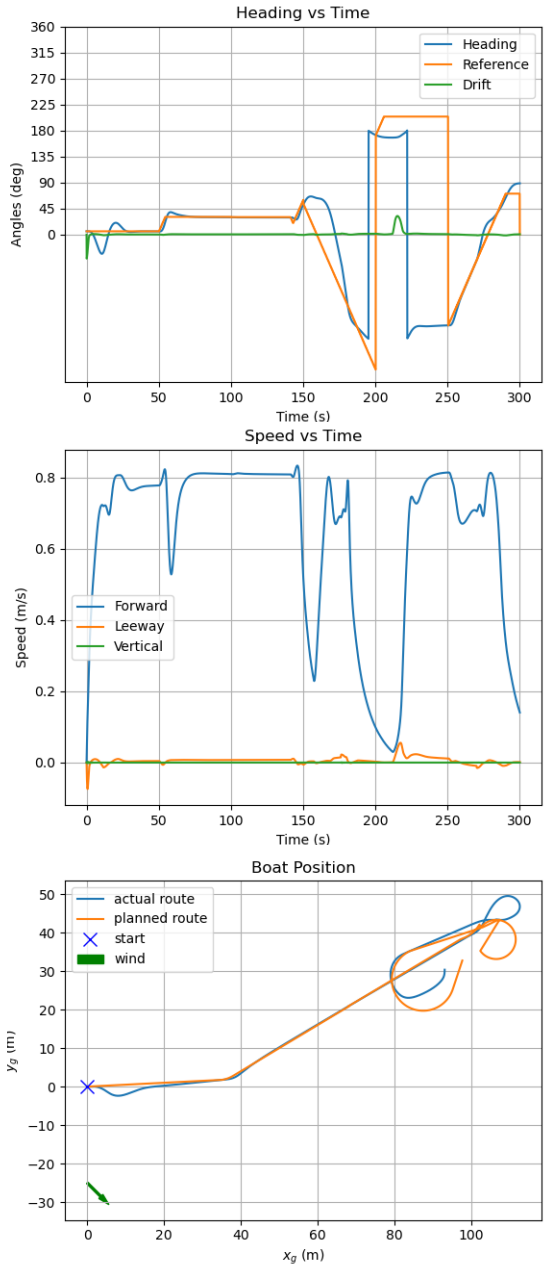

Fig. 4. Sailing with crosswinds: sailing to $(x, y) = (100, 40)$ with with crosswinds. No waves.

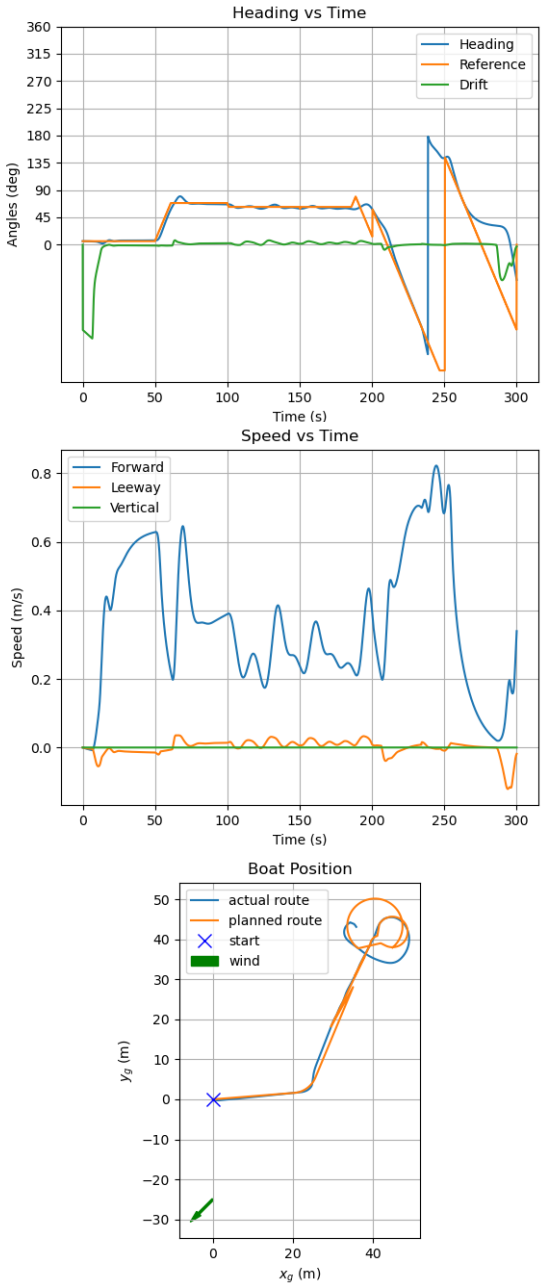

Fig. 5. Sailing against the wind: sailing to $(x, y) = (40, 40)$ with the winds against direction of sailing. No waves.

VI. Discussion

On these experiments, the boat tracks well the planned trajectory computed against the simplified boat model. The plans clearly demonstrate look-ahead capability in the experiments with obstacles, and also in planning loops around the target position. Note that the boat model is not drift free, so the planner must plan a trajectory which keeps the boat near the target once the boat reaches the target.

However, as can be seen in Figure 5 where the planned route in orange jogs back several meters around $(x, y) = (30, 20)$, the constant speed assumption of the simplified boat model begins to be severely limited when facing into the wind. The boat goes slower than expected so the replanning must account for that. This demonstrates a key limitation of the simplified boat dynamics model used here, where the speed is assumed constant and independent of the relative angle between the boat heading and the wind angle.

The planner, with the parameters as given, usually completes a plan in 1-5 seconds. However, there are some cases where it can take upwards of 30 seconds. The cause

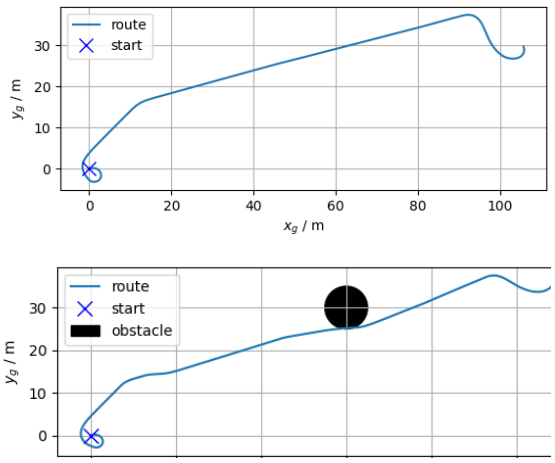

Fig. 6. Single obstacle avoidance to reach $(100, 40)$ with wind in the direction of sailing and no waves.

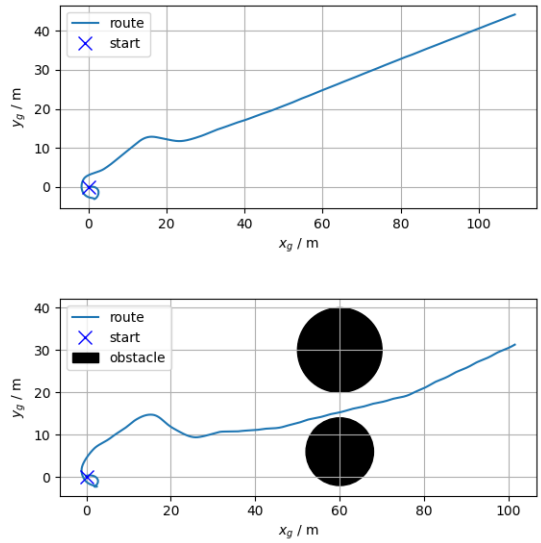

Fig. 7. Channel navigation to reach $(100, 40)$ with wind in the direction of sailing and 0.5 m waves

of these spikes in computation time is currently unknown.

VII. Conclusion

We have demonstrated a method for path planning and obstacle avoidance of a sailboat using an optimization-based planner that leverages existing heading controllers. The planner is tested in simulation, and is shown to flexible enough to allow the sailboat to traverse a channel and avoid obstacles under different environmental conditions, such as different wind directions and the presence of waves.

Although we used the stda—sailboat—simulator for simplicity, we encountered several of its limitations. The simulator uses a constant wind vector, so we were unable to test gusts of wind. Robustness to extreme weather, such as high winds, would be another goal. Successfully traversing a narrow channel with high winds would be challenging, especially if the wind were not steady.

The key limitation of this work is in the simplified boat dynamics model used by the planner. Improvements to this model to better incorporate wind would likely significantly improve performance when planning a route to sail against the wind. One such improvement might be to include the velocities in x and y directions as state variables and relate the accelerations to a cardioid with the angle being the deviation from true wind angle of the boat heading. A further improvement here would be to incorporate predictions of the effects of certain environmental disturbances such as waves. While not all environmental disturbances are predictable, waves are periodic and thus can likely be predicted well.

Another improvement would be to speed up the planner. Preliminary results suggest that one trivial way to do this would be to increase the step size used in the discretization of the continuous optimization problem. With sufficient speed up of the planner computation, the planner can be run more often, offsetting increased model mismatch between the simplified planner boat dynamics model and the true boat dynamics model. This would also make the combined planning and control system more robust to environmental disturbances.

The obstacles we present in this paper could be extended as well to support cases such as objects that drift in the water, and objects on a path such as other ships.

References

[1] M. C. Buehler, C. Heinz, and S. Kohaut, "Dynamic simulation model for an autonomous sailboat," CEUR Workshop Proceedings, vol. 2331, pp. 31–39, 2018.

[2] N. A. Cruz and J. C. Alves, "Auto-heading controller for an autonomous sailboat," in OCEANS'10 IEEE Sydney, OCEANSSYD 2010, 2010.

[3] H. Erckens, G. A. Büsser, C. Pradalier, and R. Y. Siegwart, "Avalon: Navigation strategy and trajectory following controller for an autonomous sailing vessel," IEEE Robotics and Automation Magazine, vol. 17, no. 1, pp. 45–54, mar 2010.

[4] L. Xiao and J. Jouffroy, "Modeling and Nonlinear Heading Control of Sailing Yachts," IEEE Journal of Oceanic Engineering, vol. 39, no. 2, pp. 256–268, apr 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6484198/

[5] D. Santos, A. Negreiros, J. Jacobo, L. Goncalves, A. G. Silva Junior, and J. M. Silva, "Gain-scheduling PID low-level control for robotic sailboats," Proceedings - 15th Latin American Robotics Symposium, 6th Brazilian Robotics Symposium and 9th Workshop on Robotics in Education, LARS/SBR/WRE 2018, pp. 118–123, 2018.

[6] H. Saoud, M. D. Hua, F. Plumet, and F. Ben Amar, "Optimal sail angle computation for an autonomous sailboat robot," Proceedings of the IEEE Conference on Decision and Control, vol. 54rd IEEE, no. Cdc, pp. 807–813, 2015.

[7] J. D. Setiawan, B. Arief Budiman, M. Ariyanto, T. Andromeda, D. Chrismianto, and M. A. Aziz, "Experimental Study on the Aerodynamic Performance of Autonomous Boat with Wind Propulsion and Solar Power," ICEVT 2019 - Proceeding: 6th International Conference on Electric Vehicular Technology 2019, pp. 213–219, 2019.

[8] D. H. Dos Santos and L. M. Garcia Goncalves, "Performance Evaluation of Propulsion Control Techniques for Autonomous Sailboat," 2020 Latin American Robotics Symposium, 2020 Brazilian Symposium on Robotics and 2020 Workshop on Robotics in Education, LARS-SBR-WRE 2020, 2020.

[9] F. Plumet, H. Saoud, and Minh-Duc Hua, "Line following for an autonomous sailboat using potential fields method," in 2013 MTS/IEEE OCEANS - Bergen. IEEE, jun 2013, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/6607961/

[10] M. Paravisi, D. H. Santos, V. Jorge, G. Heck, L. Gonçalves, and A. Amory, "Unmanned Surface Vehicle Simulator with Realistic Environmental Disturbances," Sensors, vol. 19, no. 5, p. 1068, mar 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/5/1068

[11] C. Alves, A. P. Aguiar, and L. Sebastião, "SAILBOT Autonomous Marine Robot of Eolic Propulsion," Tech. Rep., 2010.

[12] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," Mathematical Programming Computation, In Press, 2018.