AROB Practical Work

Jorge Ferrer Beired, Xian Pan

***Abstract*— Autonomous sailing has many interesting applications, since covering the vast area of the oceans is not feasible by human-maned ships. This work is a first approach on how to autonomously control such vehicles.**

For testing purposes, a simulator is developed that allows integration of fluid mechanics differential equations to solve the trajectory of a sailboat. This is implemented in ROS to communicate between different nodes that allow scalability of features in the future. In this version, the communication is done between the sailboat integrator and a low-level controller.

The wind, of course, has an important role in the dynamics, and the control is done performing changes in the angle of the sail and the rudder, that are the only ways in which the dynamic of the sailboat can be intentionally influenced. We will study how the different control parameters of this controller can affect the trajectories for several simple scenarios.

I. Introduction

Sailing has been a way of navigation that has had a special importance through the history of mankind. Used for the transportation of people or goods and its commerce both through sea and rivers. Many civilizations have developed independently this technology that poses a clear advantage against the manual propulsion of rowing boats or galley ships.

The dynamic of a sailboat uses interactions with its environment to modify the state. As a broad approximation, this interactions are with the wind in the sail for propulsion and with water in the rudder for turning. Hence, its control can be rather complicated because the actions that can be performed (i.e., changing the angle of the sail and rudder) do not have a clear and direct impact on the dynamics. Autonomous sailing can have nice application for surveillance of great areas both for security, maritime rescue or obtaining sensor information for meteorological predictions.

For these reasons, we believe that automatic control of a sailboat is an interesting and challenging problem. In the following work, we are going to study and implement a simulator of the dynamic of a sailboat to which a low-level controller will be applied to solve easy trajectories.

II. Algorithm description

Dynamic Model: Unlike the models studied in class (differential drive, drones, ...) where only internal forces were considered, on the dynamic model of a sailboat is necessary to account for external forces such as the wind or the water. In the literature there are different research on models, but this project is mostly inspired by the one defined by Buehler et al. [1]

The Buehler's model define the 3D dynamics of the ship, having into account the hydro-static force and buoyancy of the boat. However, we will simplify the model, assuming that the pitch and roll as well as changes in the Z axis are negligible, which results in a 2D model with $\mathbf{x_g} = (\mathbf{x}, \mathbf{y})$ and heading angle or yaw ψ. The two actuators that can be controlled are the sail angle γ and the rudder angle θ.

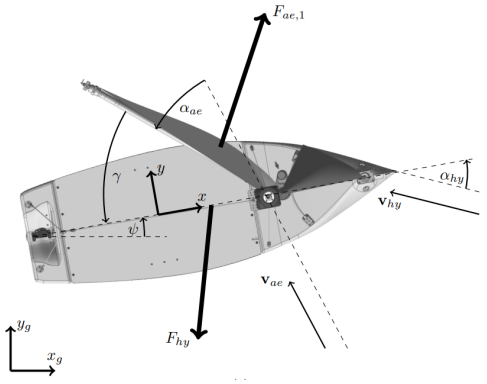


Fig. 1: Figure from Buehler et al. [1]. Top view of a sailboat. The two relevant frames are represented. The ground reference system (x_g, y_g) and the body frame defined by the $\mathbf{e_i}$ unitary coordinates. F_{ae} is the force of the sail produced by the apparent wind $\beta_{\text{WA}} = \gamma + \alpha_{ae}$, and F_{hy} is the force on the keel produced by the apparent water flow v_{hy}.

We describe the ship's dynamics as the contribution of 5 different forces: forces in the sail, the rudder, the keel and then water friction and impedance. All forces are calculated in the body frame, so its application is straightforward.

For the first three forces, elemental results from fluid mechanics for thin foils are used. Then, the force is calculated following equation 1:

$$\begin{aligned}\mathbf{F}_{\text{flow}} &= qA\left[\left(\sin\left(\beta_{WA}\right)c_L - \cos\left(\beta_{WA}\right)c_D\right)\mathbf{e_1}\right. \\ &\quad \left. - \left(\sin\left(\beta_{WA}\right)c_D + \cos\left(\beta_{WA}\right)c_L\right)\mathbf{e_2}\right] \quad (1)\end{aligned}$$

where q is the dynamic pressure, A the foil area and β_{WA} the angle of attack with the apparent flow. c_L and c_D are respectively the lift and drag coefficients, that depend on the angle of attack as defined by the original model [1].

The impedance force, or wave resistance is accounting for the limitation of speed that the ship can acquire. This resistance arise from the constructive interference between the bow and the stern waves that the boat generates on its displacement. This is approximated by equation 2

$$\mathbf{F}_{wr} = -\operatorname{sign}\left(v_x\right)c_{wr}q_{hy}A_{\text{LK}}\left(\frac{v_{hy}}{v_{hull}}\right)^4\mathbf{e_1} \quad (2)$$

where v_x is the forward velocity of the ship, c_{wr} a weight parameter predefined by the model, q_{hy} the dynamic pressure A_{LK} the lateral area, v_{hy} the current velocity and v_{hull} the maximum hull speed.

Finally a linear damping force with each of the two linear velocities and the angular yaw velocity is applied. Having all these forces into account, the change in the lineal velocity applying the chain rule on Eq. 4 is:

$$\mathbf{F} = m\frac{d}{dt}\mathbf{v} = m\left[\left(\dot{v}_x - \dot{\psi}v_y\right)\mathbf{e_1} + \left(\dot{v}_y + \dot{\psi}v_x\right)\mathbf{e_2}\right] \quad (3)$$

And the change in the angular velocity is obtained considering the inertia moments Θ produced by all the forces:

$$\tau = \frac{d}{dt}(\Theta\mathbf{\Omega}) = \dot{r}\Theta\mathbf{e_3} \quad (4)$$

The change in linear position is finally given by:

$$\frac{d}{dt}\begin{pmatrix} x_g \\ y_g \end{pmatrix} = \begin{pmatrix} v_x\cos(\psi) - v_y\sin(\psi) \\ v_x\sin(\psi) + v_y\cos(\psi) \end{pmatrix} \quad (5)$$

while the change in the heading angle is directly given by the angular velocity $\dot{\psi}$.

Control: The sailboat controlling is done through the actions of the two actuators: sail and rudder. The first one has to adjust depending on the apparent wind to try to achieve the adequate velocity for the ship. The second one is in charge of orienting the ship in the desired direction and making corrections (note that the forces in the sail produces both a displacement on the Y direction and a torque that rotates the ship). In our model, these actuators act independently and use two completely different controllers to adjust them.

The sail is controlled very simply by a controller that sets an angle that is proportional to the angle of the apparent wind [2][3]. That is to say that the sail is opened to a maximum value when the wind is in the same direction as the boat ($\beta_{\text{WA}} = 0$) and completely closed when they are in anti-parallel directions ($\beta_{\text{WA}} = \pi$). In between, the result is the linear interpolation of these two values.

$$y_s(t) = -\operatorname{sign}(\beta_{\text{WA}}) \cdot \left(\left| \frac{\beta_{\text{WA}}}{\pi} \right| \cdot \right.$$
$$\left. \cdot (\beta_{\text{WA, min}} - \beta_{\text{WA, max}}) + \beta_{\text{WA, max}} \right) \quad (6)$$

For the rudder, a PD controller is used defined as

$$y_r(t) = K_p e(t) + K_d \frac{d}{dt} e(t) \quad (7)$$

where the error is computed as $e(t) = \psi_r(t) - \psi(t)$ with $\psi_r(t)$ being the desired heading angle and $\psi(t)$ the current heading. K_p and K_d are the control parameters of the PD controller. Moreover, as it happens in the sail, the absolute value of the rudder angle is limited to a maximum value given by the design of the boat.

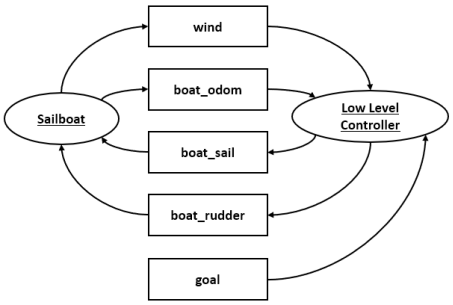


Fig. 2: System of nodes and topics in ROS designed to simulate the sailboat.

III. Implementation Details

The sailboat simulator is implemented directly in the ROS mainframe through the system of nodes and topics shown in the Fig. 2. The main node is *Sailboat*, that computes the odometry of the next state given the current dynamic state (positions and velocities), the direction and strength of the wind and the angles of the two actuators (sail and rudder). The integration is carried out using a simple first-order Euler scheme. We define a loop with a frequency of 10 Hz. At each step, the variations of both positions and velocity are computed, integrated and published on the *boat_odom* topic. The code is heavily inspired on the Buehler's Python version [1] with the simplifications exposed on Sec. II. In addition, a modification is implemented to allow actor dynamics. This takes into account that the changes in movement of the actuators are not instantaneous in real life. To change the angle of the sail, a servo can be used [4] to modify the length of a rope and trim the sail at the appropriate angle, which can take several seconds.

The other node is the *Low-level controller*. Its work is to determine the movement of the actuators given an objective and the current odometry. These two nodes communicate between themself by the pertinent topics. In *boat_odom* the odometry of the boat is published. On *wind* we publish

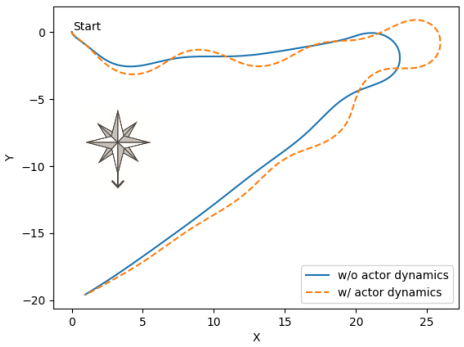
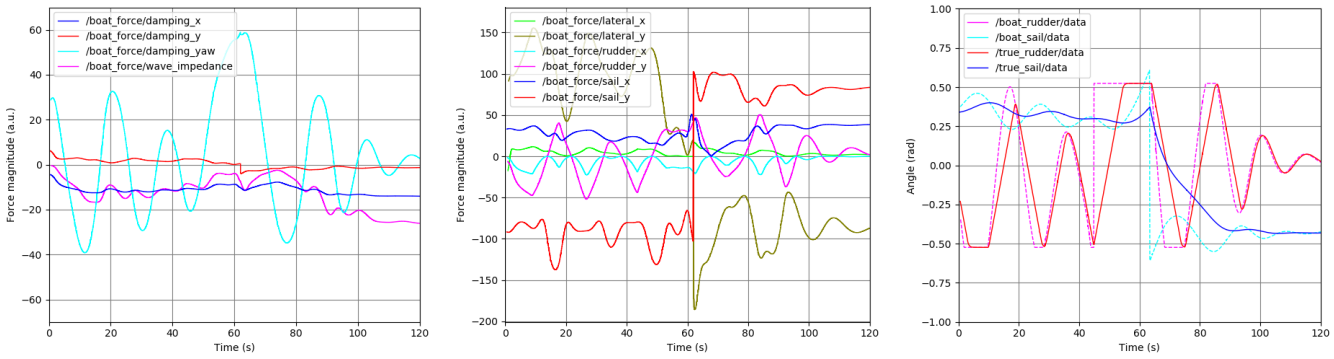


Fig. 3: Comparison between the trajectory considering the dynamics of the actuators (orange) and the trajectory without considering the dynamics (blue).



(a) Damping forces (b) Active forces (c) Rudder and sail angles

Fig. 4: Forces on the sailboat and actions of the actuators (with actor dynamics) during the trajectory described in Fig 3. Figure (a) shows the damping forces that impose on the active forces described in figure (b). In figure (c) the dashed lines represent the rudder and sail positions chosen by the LLC, and the solid lines are the actual positions. Note how the forces change direction at the second 60 when the sailboat makes the large downward turn.

the direction and strength of the wind that, for the sake of simplicity, is computed inside of the sailboat node because it is constant through a single experiment. In a future version, a new node can be used to independently publish this information if a more sophisticated behaviour was needed. Finally, the low-level controller publish the desired rudder and sail angles into the respective topics.

IV. Experimental results

Simulator validation: On Fig. 3 two trajectories are plotted using the simulator implemented. The scenario consist of a ship that starts at $\mathbf{x_g} = (\mathbf{0}, \mathbf{0})$ with the wind blowing in the direction of the negative Y. The boat has a first objective horizontally at $\mathbf{x_g} = (\mathbf{20}, \mathbf{0})$ and then has to turn $315°$ to reach the final goal at $\mathbf{x_g} = (\mathbf{0}, -\mathbf{20})$. The trajectories are controlled by the low-level controller (with $K_P = K_D = 1$). The blue trajectory corresponds to the version without actor dynamics exposed above in Sec. III and the orange one with the actor dynamics activated. With actor dynamics, the trajectory is less stable because the controller of the rudder does not take into account the delay produced by the actor dynamics. Hence, when the controller wants to make a closed turn, it can instantly set the rudder to the desired angle with actor dynamics off, otherwise it takes some time to reach the desired angle producing a larger turn (Fig. 4(c)). To make the comparisons simpler to analyse, we will work with the actor dynamics deactivated so we can certainly attribute a potential overshoot to the parameters of the controller, and not to a side effect of the actor dynamics as seen here. Several simulations for different scenarios similar to the one just depicted are run and tested against the results obtained using Buehler's code [1], that serve as a ground truth for validation.

On the physics side, note that, at the beginning, the ship goes slightly downwards instead of starting moving on a straight line. This is because the forces on the sail (Fig. 4(b)) produces a force in the side-way direction that drags the ship down. Also, the force on the sail as well as the lateral force on the keel produces moments that turn the ship clock-wise. The rudder controller tries to correct this deviation, but since the ship is moving very slowly at this point, this action is very small and produces a negligible effect. Once the ship has gained some velocity (around $x = 2.5$), the manoeuvrability is increased and the rudder is able to straighten the course of the boat.

Control Optimization: Now, we are going to perform an analysis of the different values for the parameters K_P and K_D of the PD control algorithm. For this, several simulations will be launched with the boat starting at $\mathbf{x_g} = (\mathbf{0}, \mathbf{0})$ and $\psi = 0$ (i.e., looking to the positive x) and the goal is set to $\mathbf{x_g} = (\mathbf{0}, \mathbf{30})$ so the boat has to turn to the left to reach the goal. This is graphically described on Fig. 5. Then, some combinations for the control parameters are tried for different wind directions and constant wind intensity.

For evaluating the behaviour, three cost metrics are defined:

i) Convergence time. That is just the time that the agent took to reach the goal point.

ii) Integrated heading error. Integrated error of the heading direction computed as the difference from the heading of the boat to the desired heading (defined as a straight line from the boat to the goal at each moment).

$$\int_0^{t_{\text{end}}} |\psi_d - \psi| \ \mathrm{d}t \tag{8}$$

iii) Integrated rudder action. Integrated displacement of the rudder over time. This is important because it gives a sense of the energy that is used in controlling the boat. For some applications it might be of critical importance that this value remains low.

$$\int_0^{t_{\text{end}}} |\delta\theta_{\text{rudder}}| \ \mathrm{d}t \tag{9}$$

Attending to these different metrics, we can measure which control parameters values are more desirable. There is no single pair of parameters in the sub-set tested that minimizes all three metrics at the same time for a given wind direction, and then, a compromise should be taken.

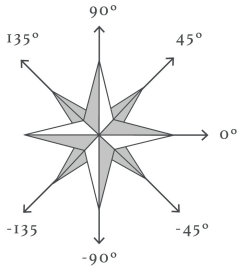


Fig. 5: Wind directions tested in the parameter space exploration of the controller.

On Tab. I there are shown the best parameters found for K_P and K_D to minimize each of the three metrics (in blackface) for the different wind directions. K_P is evaluated on the interval $[1, 0.5, 1, 2, 4]$ and K_D on $[0, 0.5, 1, 5, 10]$. All the possible combinations are tried but we can not talk about having found the "optimal" coefficients, because of our limited sub-set and arbitrary range. This range have been chosen with a previous manual study of the problem for some cases, but it is clear that if the optimal values were desired, a more sophisticated approach should be taken.

However, this can be difficult, because the best values change very broadly depending on the wind direction. This is probably one of the keys of the result of the project. The external conditions affect very dramatically to the sailboat's dynamics and it is not possible to find a single pair of values for the coefficients of the controller that perform well in all situations. As suggested by some works [5], one approach to tackle this is to have a look-up table with different coefficients for different external conditions and modify the behaviour of the controller according to this. Other authors [2][3] prefer to differentiate critical situations (for example changing directions when going against or in favour of the wind) and hardcore specific behaviours in those scenarios.

Angle	K_P	K_D	Conv. Time (s)	ψ_{error}	$\delta\theta_{\text{rudder}}$
135º	1.0	5.0	**55.40**	9.87	0.36
	4.0	10.0	55.50	**9.05**	0.38
	0.5	1.0	55.60	11.82	**0.31**
90º	2.0	5.0	**51.40**	11.22	**0.26**
	4.0	5.0	51.70	**10.65**	0.37
45º	0.5	1.0	**55.70**	15.17	0.30
	4.0	5.0	56.50	**12.36**	0.40
	1.0	1.0	56.30	14.39	**0.26**
0º	2.0	5.0	**63.10**	14.58	0.33
	4.0	5.0	64.30	**13.52**	0.43
	0.5	5.0	71.30	34.59	**0.30**
-45º	4.0	5.0	**67.20**	**25.73**	0.36
	1.0	5.0	83.40	39.11	**0.31**
-90º	1.0	10.0	**94.60**	**52.35**	0.37
	0.5	5.0	96.60	62.15	**0.34**
-135º	4.0	5.0	**109.10**	66.07	0.45
	4.0	10.0	109.60	**64.75**	0.40
	2.0	10.0	114.40	74.47	**0.36**

TABLE I: Best parameters obtained for different wind conditions, and based on different metrics.

On a broad analysis of Tab. I, we can see that $\delta\theta_{\text{rudder}}$ remain small when K_P is small, which is reasonable since this decrease the overshoot of the maneuvers, which implies that less corrections are needed. For reducing the ψ_{error} generally a high K_P is required so the correction is made as fast as possible and the ship does not stay out of course while maneuvering for a long time. It is needed to pair this value with an also high K_D in order to avoid overshooting.

Finally, if the convergence time is to be minimized, the solution is very specific in regard of the wind direction. In the simple scenario explored, the key is to make the turn upwards while loosing the least velocity doing so (huge rudder angles carry out a high turning rate, but also a significant slow-down effect) and maybe more importantly, ending up with a favorable heading direction in relation to the goal that can exploit the apparent wind direction. This is a complex problem in which it is difficult to extract clear conclusions, but we can see that the range of values preferred varies with the environment situation. For favorable wind directions, the values are generally smaller than with headwinds.

V. Conclusions

In the present work, we have studied the underlying physics of a sailboat dynamics and successfully simulate it within the ROS environment. This allows for a complex dynamical model that has given rise to interesting non-linear behaviours. As seen through the work, the control parameter space is non-trivial. The agent is dependant on the external forces and it has to control the sail and rudder angles to try to channel those environmental forces into the desired action. The environment information is no longer merely a perturbation or a tool to avoid obstacles, but now it is also a critical factor to move the boat. Understanding the forces and the effect that they can have on the sailboat has been a process that helped us later to identify the performance of the controller in the analysis phase.

The attempt to optimize the PD controller has shown that the underlying parameter space is very complex and that changing only one environmental parameter (wind angle) has a dramatic effect on it. Most on the works consulted mentioned this fact, but did not present data that supported this claim or that justified the choosing of the parameters of their model. In some cases, they just concluded that the parameters are not optimal but just "good enough". This modest work, of course, does not solve the question of the optimal way to control a sailboat, but shows that it is, in fact, a difficult problem that can not be approached carelessly.

References

[1] M. Buehler, C. Heinz, and S. Kohaut, "Dynamic simulation model for an autonomous sailboat," 2018.

[2] J. Melin, *Modeling, control and state-estimation for an autonomous sailboat*, 2015.

[3] H. Erckens, G.-A. Büsser, C. Pradalier, and R. Y. Siegwart, "Navigation strategy and trajectory following controller for an autonomous sailing vessel," 2010.

[4] B. Williamsz, J. Kane, R. J. Hartnett, and P. F. Swaszek, "Senior project design of a two meter autonomous sailboa," 2014.

[5] D. Santos, A. Negreiros, J. Jacobo, L. Goncalves, A. Silva Junior, and J. M. Silva, "Gain-scheduling pid low-level control for robotic sailboats," 2018.