Autonomous control of a sailboat

Neelay Junnarkar, Andrew Fearing, Hamza Khawaja

May 15, 2021

Outline

Motivation

Figure 1: Boat stuck

Why Sailboats?

Sailboats are useful for low-power long-term deployments. Also, present a gap in current literature, especially in controller design.

Control of sailboats

Controlling boats autonomously presents an interesting challenge due to the large effect of environmental disturbances such as wind, water currents, and waves.

Goal

Develop planning and control scheme which can take into account environmental disturbances and obstacles such as channels.

Sailing

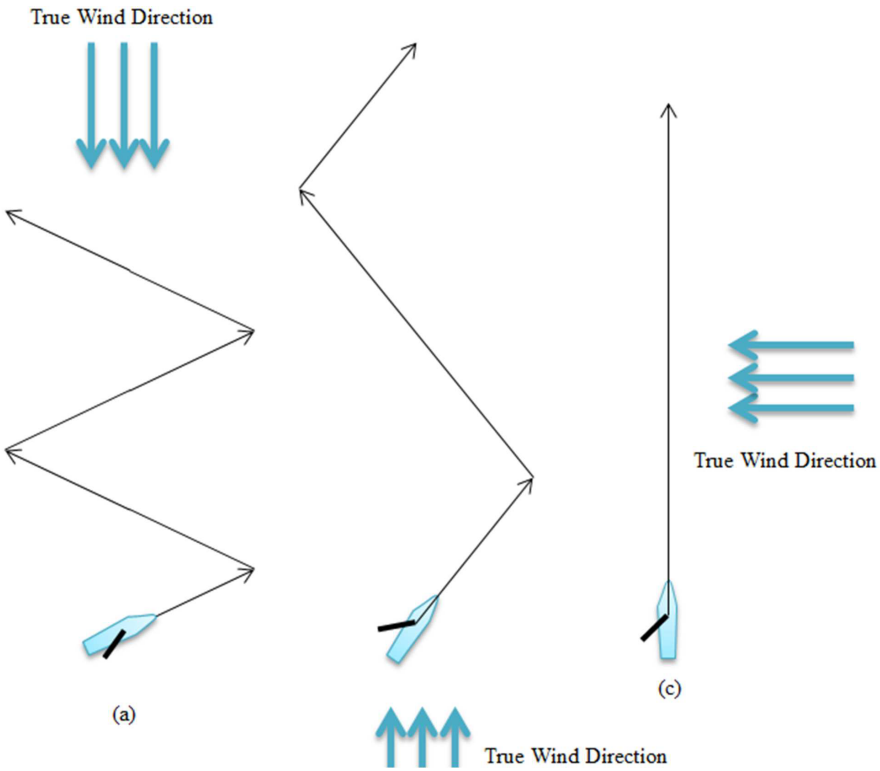


Figure 2: Modes of sailing [1]

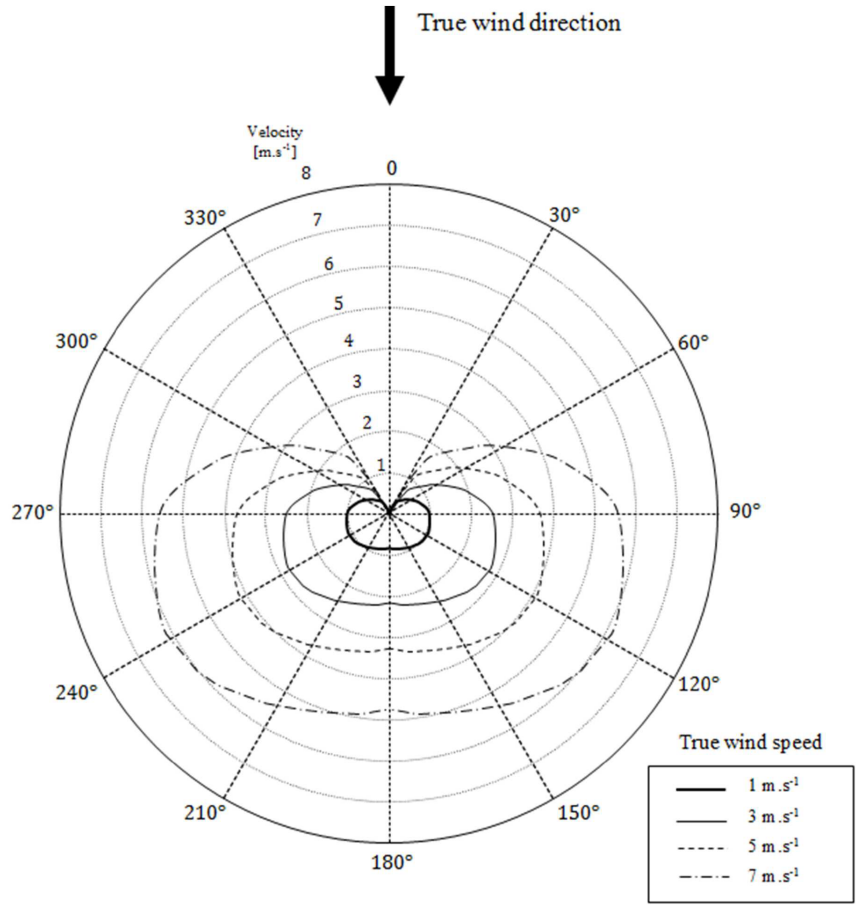


Figure 3: Velocity polar diagram [1]

Sailboats trajectories depend on wind direction

Sailboat Components

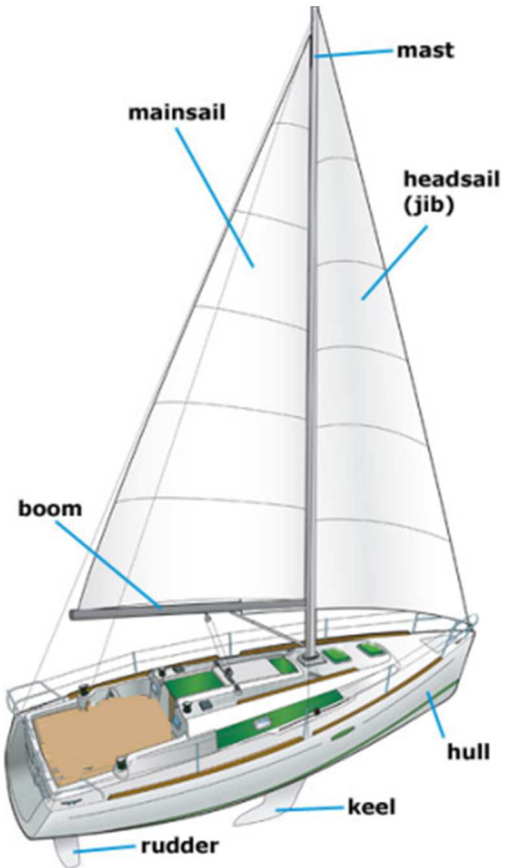


Figure 4: Sailboat main components [1].

- Note that headsail and mainsail are connected in our model for simplicity
- Two control surfaces: sail and rudder $\Longrightarrow$ two actuators

6-DoF Model Axes and State Space

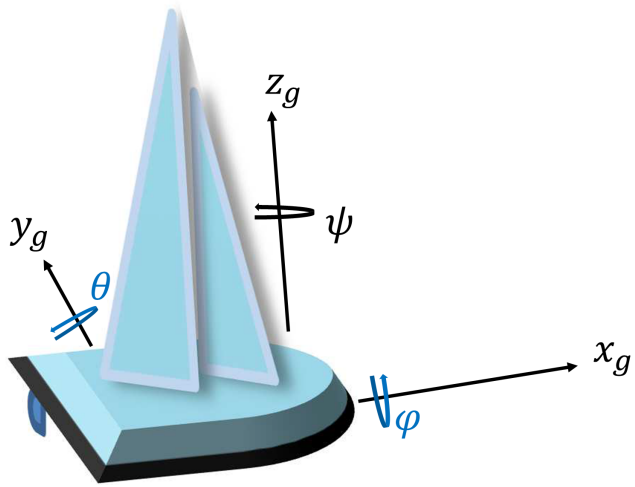


Figure 5: Sailboat axes [1]

- Global frame
- Boat frame

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 1: The 14 sailboat state variables [2].

Dynamics Model

Position:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_g \\ y_g \\ z_g \end{pmatrix} = \begin{pmatrix} v_x \cos(\psi) - v_y \sin(\psi) \\ v_x \sin(\psi) + v_y \cos(\psi) \\ v_z \end{pmatrix} \tag{1}$$

Angles:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} \varphi \\ \theta \\ \psi \end{pmatrix} \approx \begin{pmatrix} p \\ \cos(\varphi) q - \sin(\varphi) r \\ \cos(\varphi) r + \sin(\varphi) q \end{pmatrix} \tag{2}$$

Velocities:

$$m \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v} = \mathbf{F} \tag{3}$$

Angular Velocities:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\Theta\Omega) = \mathbf{T} = \sum_i r_i \times \mathbf{F}_i \tag{4}$$

Forces and torques are derived from physical interactions of environment, sailboat, sail, and rudder.

Control Surfaces

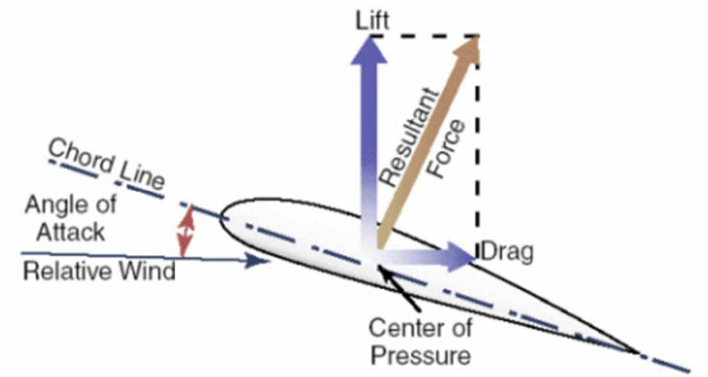


Figure 6: Drag and lift forces on a foil [1].

Because the boat is moving, foils have "apparent velocity" in fluid

$$\mathbf{V}_{af} = \mathbf{V}_f - \mathbf{V} \tag{5}$$

$\mathbf{V}_{af}$ is the apparent velocity of the foil relative to the fluid
$\mathbf{V}_f$ is the velocity of the fluid
$\mathbf{V}$ is the velocity of the sailboat

$$\mathbf{F}_{\text{flow}} = \frac{1}{2}\rho v_{rel}^2 A \begin{pmatrix} \sin(\beta)c_L - \cos(\beta)c_D \\ \sin(\beta)c_D + \cos(\beta)c_L \\ 0 \end{pmatrix} \tag{6}$$

where C_L and C_D are lift and drag coefficients, respectively; β is the apparent flow angle; A is the foil area; and ρ is the fluid mass density.

Control Surfaces, continued

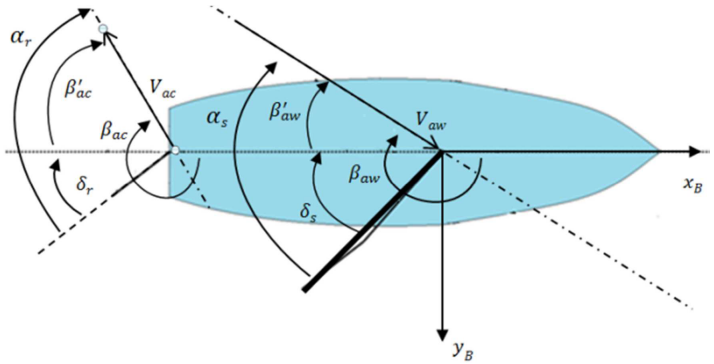


Figure 7: Control surface angles [1].

Bottom line

- Set set angle to maximize wind force – use geometry
- Rudder induces moment about z) - axis – relies on keel

$\therefore$ Only need to control rudder

Evaluating Simulators

Simulators for sailboat models are significantly underdeveloped.

`USVSim` [3]

- Ubuntu 16.04 + Kinetic ROS + Gazebo
- 6 DoF boat dynamics model.
- Waves, buoyancy, water currents, wind currents, thruster underwater, thruster above water, foil.
- Too complicated to work with, hard to extend.

`stda-sailboat-sim` [2]

- Python + Numpy + Matplotlib
- 6 DoF boat dynamics model.
- Waves, buoyancy, wind, sail, keel, rudder forces.
- Good enough .

High-level Control Design

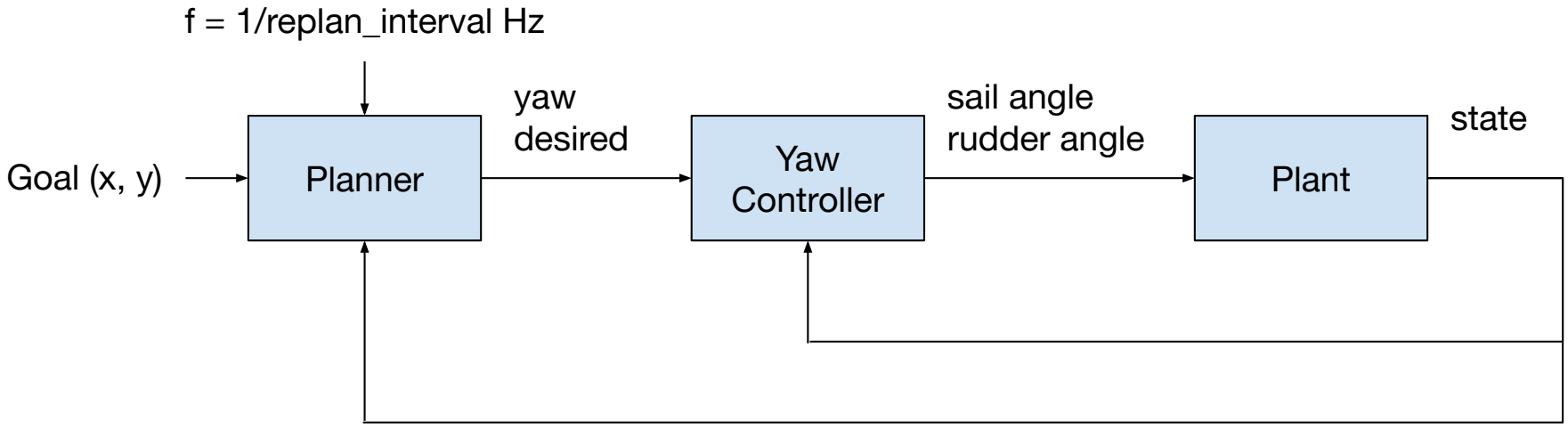


Figure 8: Block diagram of planner and controller

A planner generates a reference heading (yaw) for the boat to reach a given x, y coordinate goal and boat state. The planner is run at a low rate since planning, and also the boat itself, is slow.

The yaw controller generates sail angles and rudder angles to be used as inputs to the boat to track the reference heading.

Yaw Controller

Yaw controller tracks a reference yaw by setting sail and rudder angles.

Implemented as per the same paper which developed the simulator and boat dynamics model we are using[2].

This controller leverages the fact that there is an optimal angle for the sail based on wind direction to the rudder and sail angle control.

Note that the inputs called here as ‘rudder angle’ and ‘sail angle’ are actually inputs to a first order model that simulates delays in changing the rudder and sail angles.

Yaw Controller

Apparent wind is the wind relative to the boat (accounting for boat velocity).

With θ_{aw} being the angle of the apparent wind, sail angle is optimally set based on current boat yaw as

$$\theta_{\text{sail}} = \frac{\sin(\theta_{aw})}{\cos(\theta_{aw}) + 0.4\cos^2(\theta_{aw})} \tag{7}$$

This is clipped based on sailboat limit.

The rudder controller is implemented as a PID controller derived from a Jacobian linearization of a simplified dynamics model from input rudder to output yaw.

Gains are determined through LQR.

Planner

The planner takes in a high level x, y goal and plans a yaw trajectory for the boat to reach the goal.

We design the planner as repeatedly solving constrained nonlinear optimization problems with receding horizons. Due to model complexity, the planner runs slowly and so we define a replanning interval on the order of 10s of seconds.

Objective: the standard LQR-like quadratic objective with Q and R tunable cost matrices.

Constraints: planned path must satisfy simplified boat dynamics and avoid obstacles.

The optimization problems are solved with Casadi and the IPOPT solver.

Planner: Boat Model

A simplified boat dynamics model is used in the planner to speed up planning. Also, some information used by the boat dynamics model are not well known ahead of time.

We make a simplified 3 state boat dynamics model with states x, y, ψ where x, y are position and ψ is heading, and a non-physical input u such that

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v_x \cos(\psi) - v_y \cos(\psi) \\ v_x \sin(\psi) + v_y \cos(\psi) \\ u \end{bmatrix} \tag{8}$$

Set yaw as integral of the pseudo-input instead of setting yaw as the input itself to:

- Ensures yaw is continuous.
- Allows limiting rate of change of yaw, ensuring planned path can be tracked.

Planner: Boat Model

Limitations of this boat model:

- Velocity model is severely simplified.
  - We model v_x and v_y as constant over the planning horizon and equal to the velocities at the time of planning.
  - Attempts at more detailed models such as setting speed as a cardioid based on difference between yaw and wind direction resulted in optimization failing to solve.
- Model accuracy depends significantly on relative angle between wind and boat heading.
  - Planner sometimes outputs bad paths when boat must go against wind.

Replanning handles deviations from plan due to model mismatch and disturbances.

Planner: Obstacles

We utilize convex casadi constraints to model ”spherical” obstacles. Each circle with x-position, x_obj and y-position, y_{obj} and radius, r_{obj}. Then the constraint equation for each obstacle, where :

$$(x - x_{obj})^2 + (y - y_{obj})^2 >= r_{obj}^2 \tag{9}$$

The radius r_{obj} used above is more than the actual radius of the obstacle to avoid collision.

The modeling makes creating objects easier than other polygons and custom shapes. Some real-life objects that sail boats can interact with, e.g buoys, can be modeled using ”spheres”

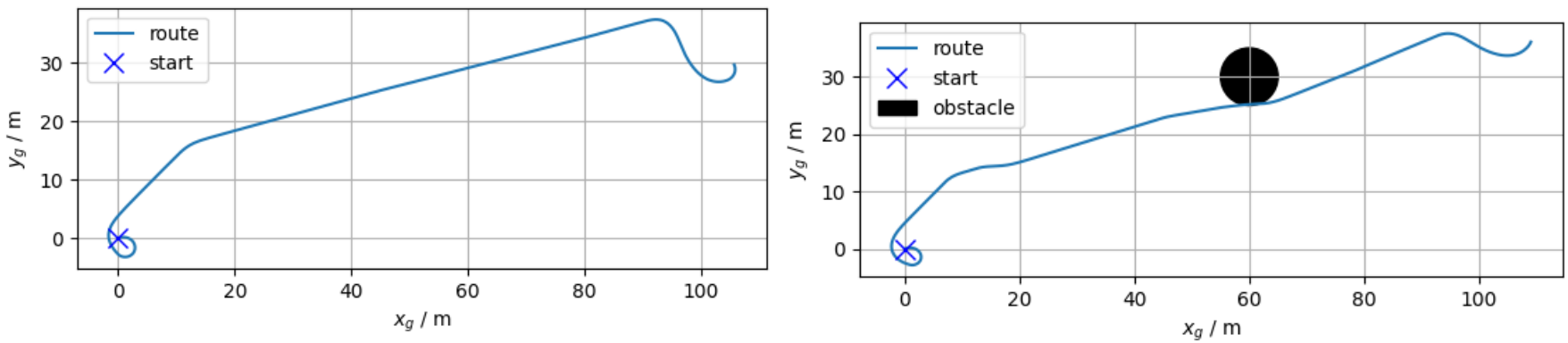
Planner: Obstacles

Limitations of this Obstacle model:

- Obstacle model is simplified.
  - Modeling real-life objects as circles can be inaccurate, especially when the shapes are highly non-convex.
  - Trying to emulate different shapes using circles increases computation time and can be infeasible if a large number of objects are being considered.
- Obstacle model is not completely practical
  - The obstacles modelled are stationary, which is usually not the case in dynamic oceanic environments

Experiments: Obstacle planning, no waves

The following experiments were run with a planning horizon of 50 seconds and a replanning interval of 50 seconds. Positions are plotted below.

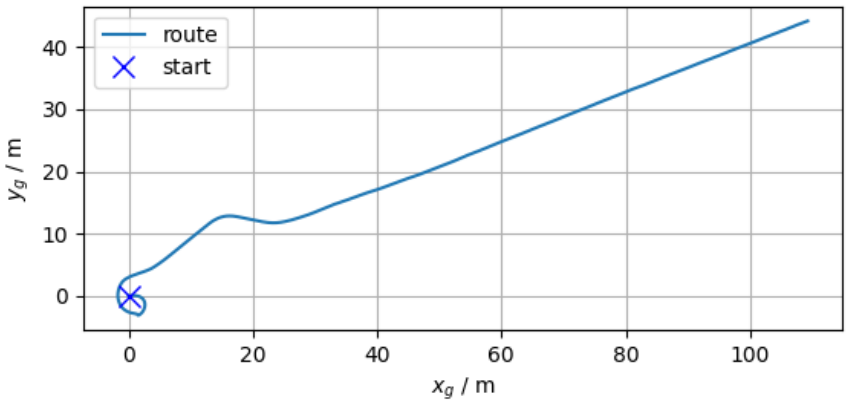


(a) No obstacle

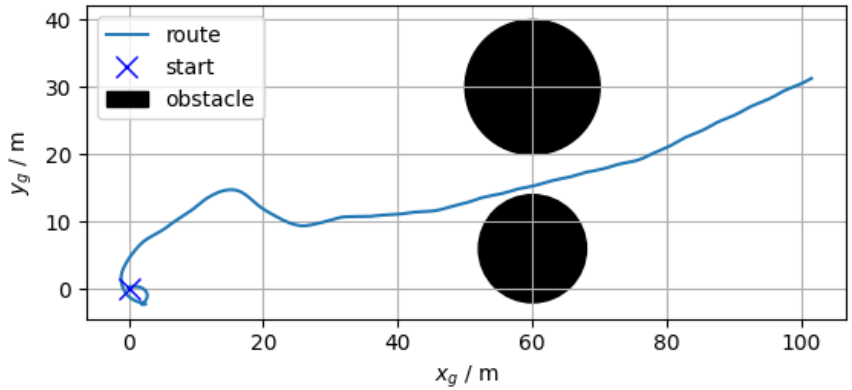
(b) One obstacle

Figure 9: Sailing to $(x, y) = (100, 40)$ with the wind in direction of sailing. No waves.

Experiment: Obstacles with waves



(a) No obstacles



(b) Two obstacles

Figure 10: Sailing to $(x, y) = (100, 40)$ with the wind in direction of sailing. 0.5m Waves included.

Experiment: Sailing with Wind

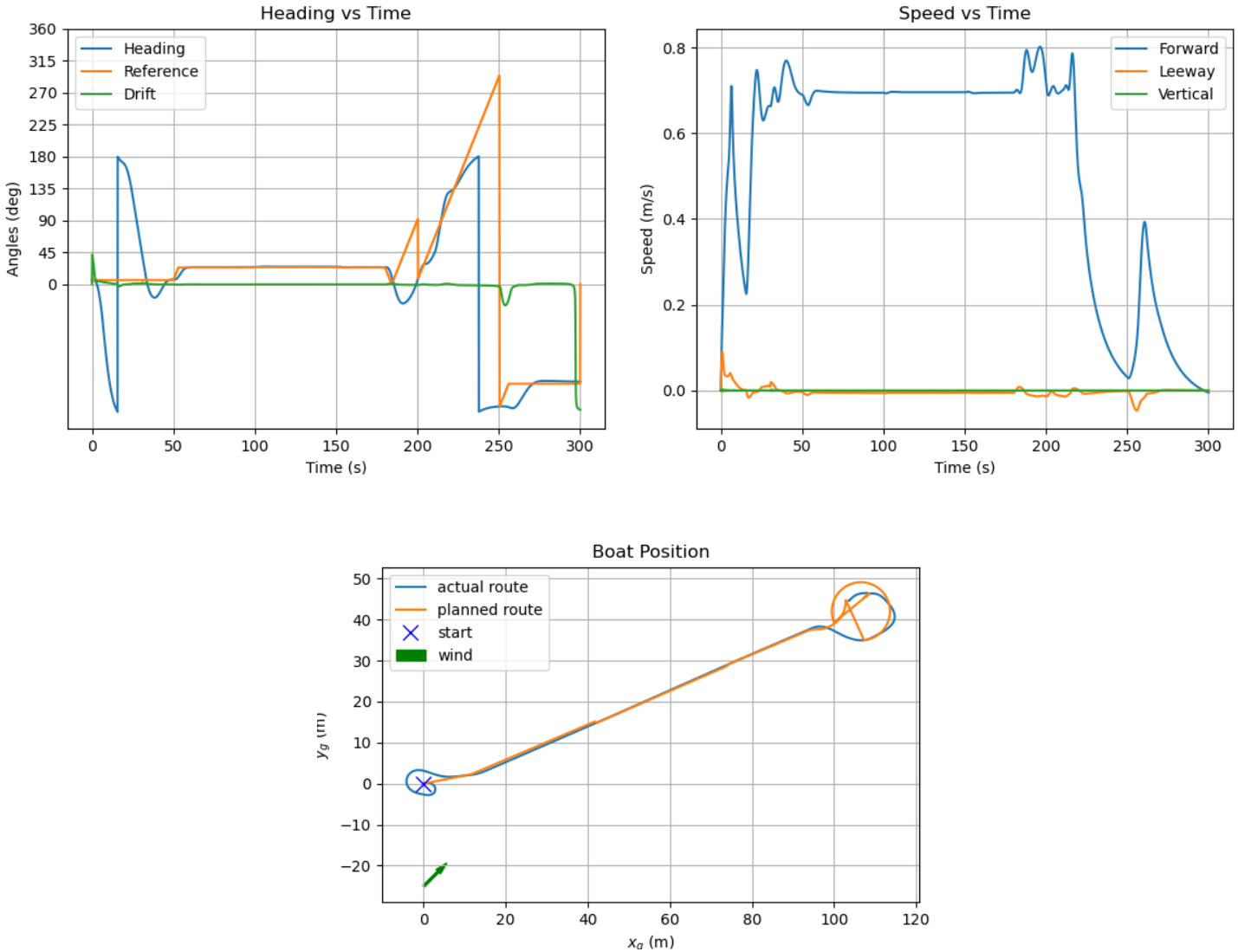


Figure 11: Sailing to $(x, y) = (100, 40)$ with the wind in direction of sailing. No waves.

Experiment: Sailing with Crosswinds

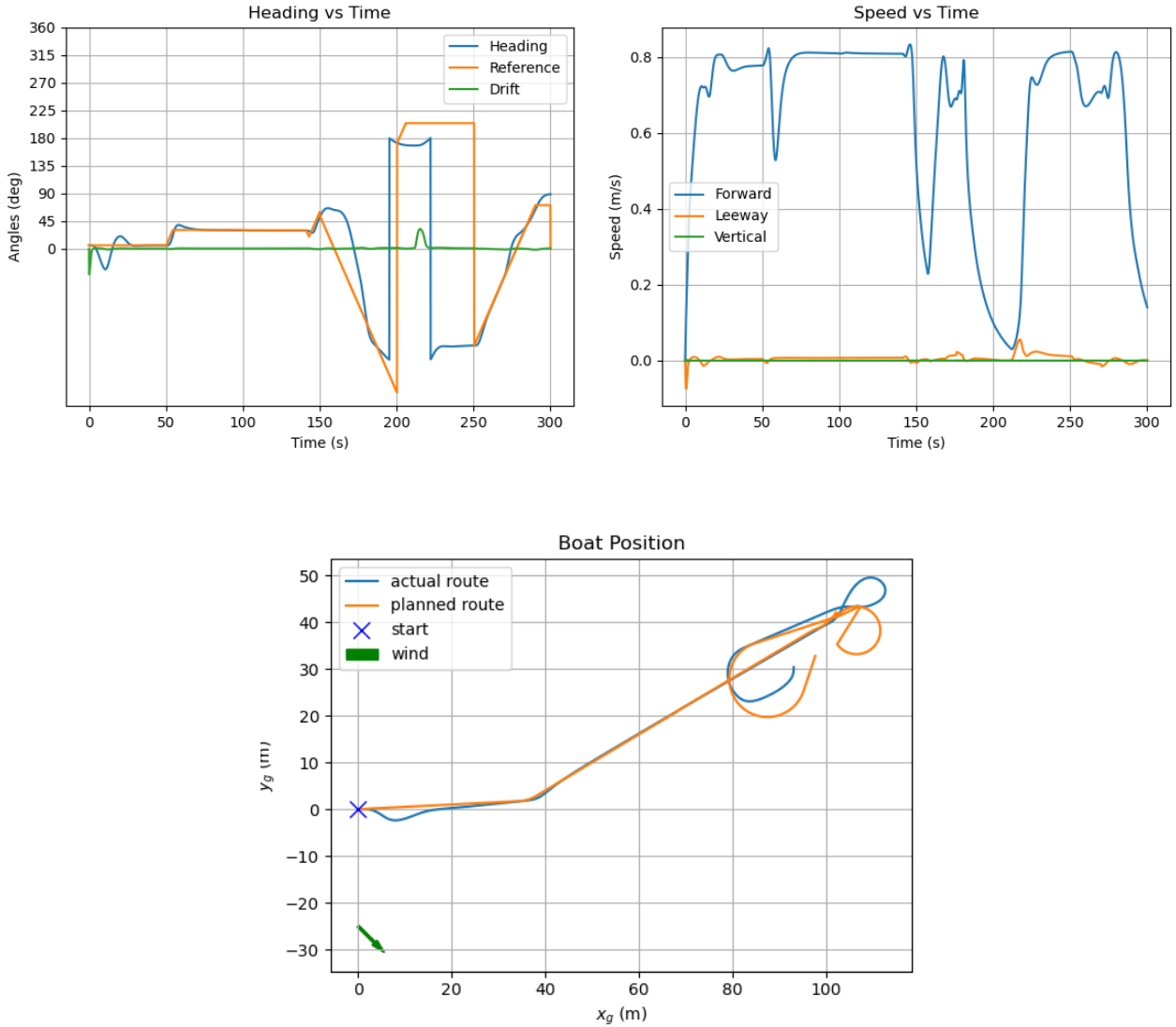


Figure 12: Sailing to $(x, y) = (100, 40)$ with the winds cross to direction of sailing. No waves.

Experiment: Sailing Against Wind

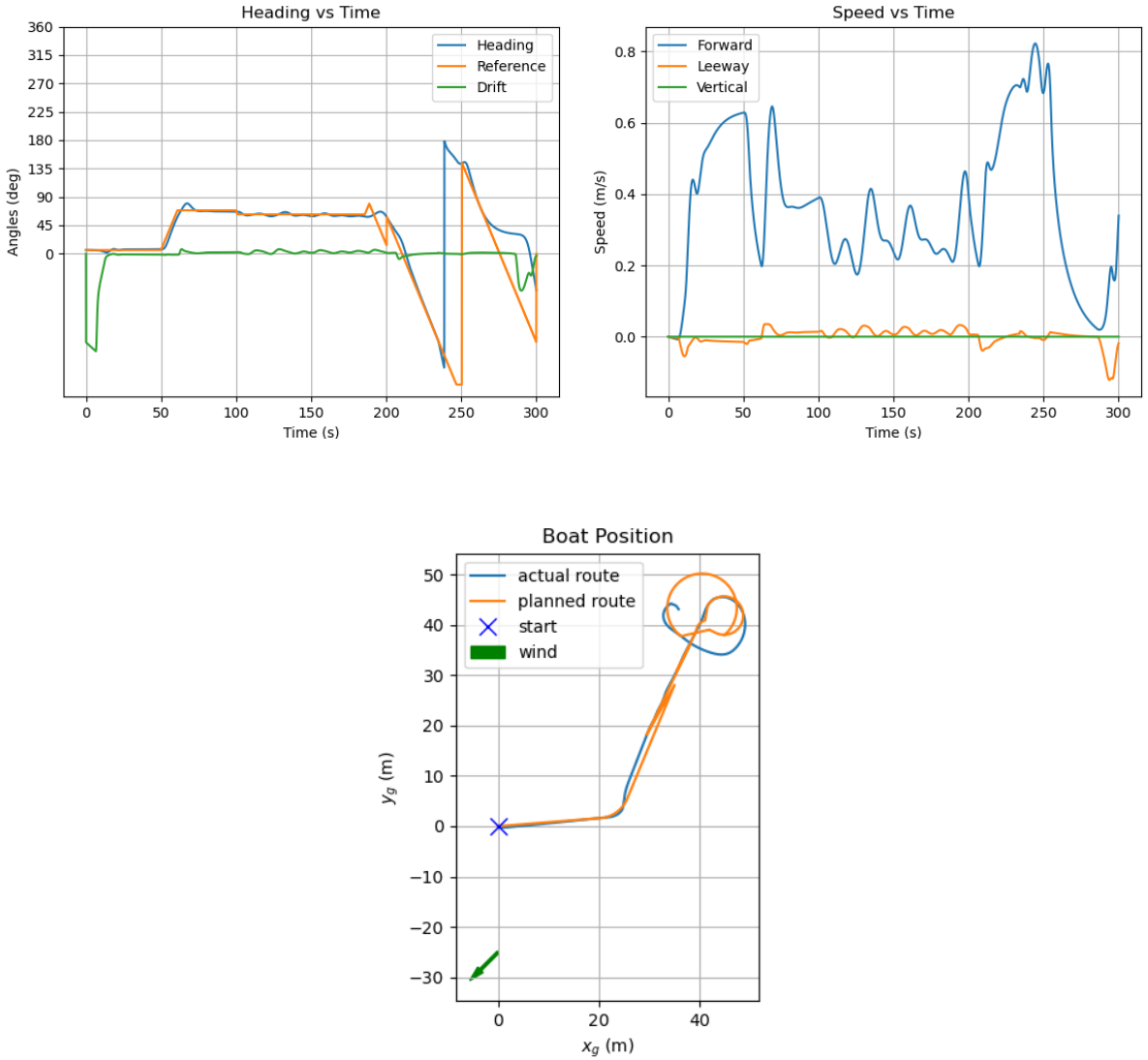


Figure 13: Sailing to $(x, y) = (40, 40)$ with the winds against direction of sailing. No waves.

Future Work and Improvements

- Improve simplified planner boat dynamics model:
  - Speed up planner.
  - More accurately model boat velocity.
  - More accurately model or predict environment (to better handle environmental disturbances).
- Expand obstacle model to support:
  - More complex geometries.
  - Simple dynamics like drifting in water.

Conclusion

In our project we develop and present an optimization-based planner for generating reference headings from high-level position targets, to pass to yaw controllers, which are more developed in literature.

The optimization-based planner is flexible enough to handle additional constraints such as obstacle avoidance, and is fast enough to allow for replanning to create a closed-loop planning and control system.

Modular design where planner and yaw controller blocks can be switched out.

Sailboat simulator library which is modular and usable for testing planners and controllers.

Thank You

Questions?

References

- Carlos Alves, A Pedro Aguiar, and Luis Sebastião. *SAILBOT Autonomous Marine Robot of Eolic Propulsion*. Tech. rep. 2010.
- Moritz C. Buehler, Carsten Heinz, and Simon Kohaut. “Dynamic simulation model for an autonomous sailboat”. In: *CEUR Workshop Proceedings* 2331 (2018), pp. 31–39. ISSN: 16130073.
- Marcelo Paravisi et al. “Unmanned Surface Vehicle Simulator with Realistic Environmental Disturbances”. In: *Sensors* 19.5 (Mar. 2019), p. 1068. ISSN: 1424-8220. DOI: 10.3390/s19051068. URL: https://www.mdpi.com/1424-8220/19/5/1068.