MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Deep Reinforcement Learning for Optimal Sailing Upwind

Suda, Takumi; Nikovski, Daniel

TR2022-102 September 17, 2022

Abstract

We describe the application of deep reinforcement learning (DRL) methods to determine the optimal decision policy when sailing a sailboat towards a target point located upwind from the boat's current position, under the conditions of wind direction and speed that vary according to an unknown stochastic process, as is typical in real sailing races. A model of the dynamics of the sailboat is described together with a suitable choice of actions, in the form of a Markov decision process (MDP), which allows the application of a wide variety of DRL algorithms. Empirical results show that the learned policy outperforms baseline control algorithms that do not take into consideration the variability in wind strength and direction, and instead assume that the current wind conditions will persist indefinitel

IEEE International Joint Conference on Neural Networks IJCNN 2022

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Deep Reinforcement Learning for Optimal Sailing Upwind

Takumi Suda
Mitsubishi Electric Research Labs
Cambridge, Massachusetts, 02139
email: suda@merl.com

Daniel Nikovski
Mitsubishi Electric Research Labs
Cambridge, Massachusetts, 02139
email: nikovski@merl.com

***Abstract*—We describe the application of deep reinforcement learning (DRL) methods to determine the optimal decision policy when sailing a sailboat towards a target point located upwind from the boat's current position, under the conditions of wind direction and speed that vary according to an unknown stochastic process, as is typical in real sailing races. A model of the dynamics of the sailboat is described together with a suitable choice of actions, in the form of a Markov decision process (MDP), which allows the application of a wide variety of DRL algorithms. Empirical results show that the learned policy outperforms baseline control algorithms that do not take into consideration the variability in wind strength and direction, and instead assume that the current wind conditions will persist indefinitely.**

***Index Terms*—Optimal control under uncertainty, stochastic modeling**

I. Introduction

Sailing a boat towards a point upwind in a minimum time is a difficult optimal control problem due to the limitations to the motion of the boat and the generally stochastic nature of the velocity and direction of the wind that powers that motion [1]–[3]. In the situation displayed in Fig. 1, typical of the first and often subsequent legs of sailing races, the boat starts from a position that is directly downwind from the location of an upwind buoy, also called the windward mark, and its objective is to reach and sail around that buoy in a specified direction, usually counter-clockwise. As sailing directly upwind is impossible for sailcraft, reaching the buoy would require a series of maneuvers, where the boat sails as close to the wind as possible without stalling or losing speed, and changing its direction of motion at least once. It is the job of the skipper of the boat to decide at what angle to the wind to sail, as well as when to change directions (tack). This problem is exacerbated by the generally variable direction of the wind, which the skipper can observe and possibly predict, based on past observations. A reasonable assumption is that within the duration of a single leg of a sailing race, the wind direction and speed will not be constant, but the stochastic process that generates them will be constant, although generally unknown. (The dominant direction of the wind is usually known with some accuracy, as it determines the relative position of the buoy and the start line of the race.) The uncertainty in the wind direction (and to a lesser extend, speed) affects significantly the decisions of the skipper. Although it is generally possible to reach the goal mark with a single tack (blue trajectory in Fig. 1), if the skipper tacks too early (red trajectory), the boat might miss the mark, necessitating two more tacks to recover. Each tack slows down the motion of the boat, because its bow must go through the direction of the wind, at minimal speed, thus losing time. On the other hand, if the skipper tacks too late (green trajectory in Fig. 1), the boat will travel an unnecessarily long distance, also losing some time.

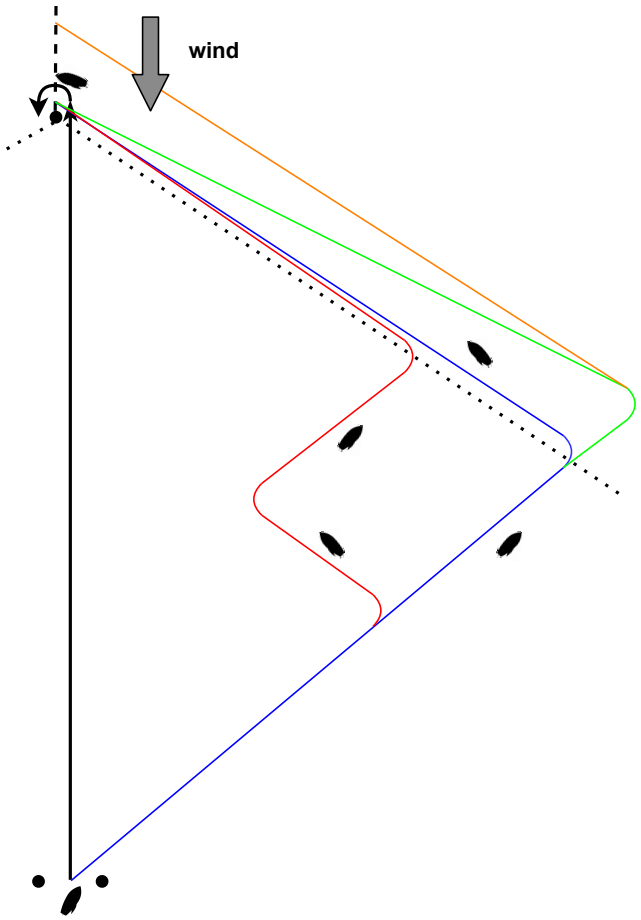

Fig. 1. Sailing course. The boat starts from a point directly downwind, and must reach and round the upwind mark in a counter-clockwise direction by executing a series of maneuvers.

The optimal control policy needed by the skipper will consist of a mapping from the relative position of the boat with

respect to the goal mark and the current direction of the wind to the desired point of sail (heading of the boat) and the associated choice of whether to tack or stay on the current course. The sequential nature of this decision problem, the hybrid form of its decision variables, as well as its stochasticity, make it very difficult to solve by means of classical optimal control techniques, and suggest that it might be a good target for solution by means of deep reinforcement learning algorithms [4]. We present one such solution in the next sections. Section II reviews several already proposed solutions to the problem of optimal sailing upwind. Section III defines the particular equations that we have chosen to model the motion of the boat in variable wind conditions, and Section IV describes how these equations, along with a suitable stochastic model for the direction and speed of the wind, can be represented as one of two suitable Markov decision processes (MDPs). Section V presents empirical results from the application of several well-known deep reinforcement learning algorithms to the two MDPs, and compares them to two baseline model-predictive controllers that do not take into consideration the variability of the wind. Section VI concludes and proposes several possible directions for future work.

II. Related Work

Sailcraft sailing skills have been perfected over the centuries, and are usually acquired by human sailors over a long period of training and practice. In addition, recent advances in computational technology have led to the application of advanced control and optimization algorithms to various aspects of sailboat steering and navigation [5]. The basic racing technique when sailing upwind usually consists of the following elements. At any given moment, the highest-level decision the skipper must make is whether to stay on the current tack (port or starboard, meaning the wind comes over the left or right board of the boat, respectively), or change tack. Depending on this decision (change tack or stay the course), the skipper must then determine what the desired new heading of the boat should be. This is an intermediate-level decision, also called a helming decision. Once the desired new heading has been determined, the skipper must make the low-level decision of what rudder angle and sail(s) position to choose in order to get to the desired heading optimally. In the case when the high-level decision has been to change tack, the desired new heading will generally be very different from the current heading, so a complex tacking maneuver might have to be executed that requires exact coordination between the rudder and sails of the boat. These three levels of decisions are usually well separated, and on a large sailing boat might be made and executed by three different sailors (certainly, in coordination with one another). Similarly, on robotic sailing boats, they are usually separated into three different layers of the control system architecture [6], [7].

The lowest-level control of the rudder and sails can usually be implemented by traditional control systems technology, such as a proportional-derivative-integral (PID) controller for the rudder angle, and a look-up table for the optimal trim of the sails given a particular point of sail with respect to the wind [5]. As such, these decisions are easy to automate, usually in a boat-specific manner. The next level up, the helming decision about what angle with respect to the wind to choose, has been researched extensively by computational methods. Algorithms and software that predict a particular boat's speed given the wind's speed and angle with respect to the boat's current heading are called Velocity Prediction Programs (VPPs) [8]. VPPs have been used in boat design and racing for a number of decades, and are part of the instrumentation of many modern sailboats. Their output can be summarized in a so-called polar diagram, an example of which is shown in Fig. 2. This polar diagram, specific for each boat, reflects what the surface speed of the boat will be if its heading is at an angle α with respect to the true wind angle (TWA), and the current true wind speed (TWS) is h. Fig. 2 shows this dependency for one specific TWS. This speed is achievable under the assumption that the sails are trimmed (adjusted) optimally with respect to the wind's direction by the lower-level controller. With the help of the polar diagram, the optimal heading of the boat when sailing upwind can be determined by projecting the boat's velocity on the vertical axis, and choosing the angle α^* corresponding to the largest projection, as shown in Fig. 2. By maximizing this projection, also called Velocity Made Good (VMG), the boat will make maximal progress against the wind, which is the main objective when sailing upwind. So, if a polar diagram of a boat is available, the intermediate helming decision can be made with its help. One additional circumstance in this calculation is whether the boat can reach the mark on the current tack, or not. If it is not possible, the course maximizing VMG is optimal. If, on the contrary, it is possible to reach the mark, it is usually more advantageous to head directly for the mark, as shown in the green trajectory in Fig. 1. However, because an unfavorable later wind change might make the boat miss the mark, unless it tacks two more times to gain distance against the wind, it might still make sense to head to a point windward of the mark, as a precaution.

To address the highest level of decision making, the course planning part of the control problem, a number of computational methods have been applied [5], [7]. Due to its computational difficulty, this part of the problem is still far from being solved optimally. The most important complicating factor is the variability in the wind's direction and speed. If there were no such variability, the optimal path can be computed easily by sailing consecutively on two opposite tacks at the optimal angle to the wind, and transitioning between the two by a single tacking maneuver, as shown in the blue trajectory on Fig. 1. For the situation shown there, because the mark needs to be rounded counter-clockwise, the boat will need to be on a starboard tack when reaching it (that is, wind coming over its right board). This means that it will have to sail initially on the opposite, port tack (wind coming over its left board), and tack to starboard at the right time. If the optimal angle for sailing upwind is α^*, the right moment to change tacks is when the boat crosses the straight line extending from the mark at angle of α^* to the TWD. This line is called the

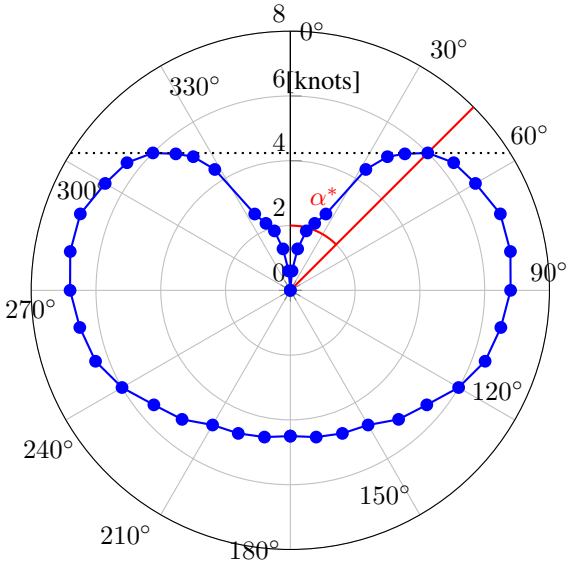

Fig. 2. Polar diagram.

layline, and is shown with a dotted line in Fig. 1. Stelzer and Pröll describe the implementation of this sailing strategy on a robotic boat, also accounting for the downwind drift the boat will experience when sailing upwind [9].

However, when the wind direction shifts, the layline will shift with it, and if the boat has already tacked based on the previous layline, it might end up on a starboard point of sail that will no longer reach the mark. In this case, the boat will have to tack two more times to recover and regain distance upwind. This sequence is shown in the red trajectory in Fig. 1, for a boat that has already tacked, maybe because the direction of the wind at a previous time was such that the boat had crossed the layline at the time.

How to make the optimal decision about when to tack is thus a difficult sequential decision problem that has been addressed by various optimization methods. Philpott and Mason formulated the problem of optimizing sailboat routes under uncertainty as a sequential decision making problem, and proposed a solution method based on dynamic programming (DP) [1]. They proposed a mean-reversal stochastic model for wind speed and direction. However, because they employed DP with a discrete state space, and arranged the discrete states in stages according to their distance upwind to the mark, they made several simplifications that might introduce inaccuracies. First, they assumed that transitioning between stages takes approximately a constant time, but this transition time clearly depends on the speed of the boat and is variable. Furthermore, this assumption did not let them account correctly for the time it takes to execute a tacking maneuver, when the boat slows down dramatically, so they had to add an extra penalty time for each tack. In addition, they had to discretize the direction of the wind into a relatively small number of bins, essentially modeling it as a Markov chain. Although constrained by the limitations imposed by the solution method (discrete-space DP), their work was an early success of the sequential decision making approach to solving the route planning problem for sailboats. This work also pioneered the idea that the result of the computation should not be merely a single optimal trajectory, but an entire decision policy that can specify the optimal action for every state the boat might find itself in.

Later on, Ferguson and Elinas built upon this work, modeling the problem as a Markov decision problem (MDP), but in order to make its solution tractable, they discretized its state space, too [2]. Moreover, they did not include the heading angle of the boat in the state space, and instead assumed that the boat could only be on exact close-hauled starboard or port tacks (at angles $+\alpha^*$ or $-\alpha^*$ to the TWD, respectively). Because the intermediate headings when tacking were not included in the state space, it was assumed that the tacking maneuver was performed in a single step, and a time penalty for it was added to the cost function of the MDP. As the wind shifts stochastically, the transitions of the MDP become stochastic, too, which is reflected in the transition probabilities of the MDP. The optimal policy for the resulting MDP was computed by means of the value iteration (VI) algorithm [10]. Moreover, they allowed the MDP to have only two possible actions: change tacks and stay on the current tack, respectively. While this helps significantly as regards the computational speed of the VI solver, it probably results in a sub-optimality of the computed policy – as pointed above, sailing close-hauled at an angle α^* to the TWD is not necessarily always optimal on the final starboard approach to the mark, where sailing directly to the mark would be faster.

Feretti and Festa [3] formulated the route planning problem as a hybrid optimal control problem, including in the state vector of the system the tack of the boat (a discrete variable) along with its coordinates (continuous variables). Wind direction and speed were modeled by means of stochastic differential equations (SDE). The resulting Hamilton-Jacobi-Bellman (HJB) equations of optimal control were solved numerically on a grid of discrete nodes in space and time. A modified policy iteration scheme was used to speed up the computation.

An additional aspect of route planning in sailboat racing is the risk attitude of the skipper of the boat. Whereas the usual sailing objective is to minimize the expected (average) time to the mark, corresponding to a risk-neutral attitude towards the actual probabilistic distribution of that time, in real races, the risk attitude of skippers would change depending on their relative position in the race. Skippers that are lagging would usually try to choose a risky course with high variance of time to mark in the hope that that would give them a chance to catch up with the leaders and possibly win the race, whereas skippers in the lead would rather choose a conservative strategy that would help them preserve their lead with maximal probability. Tagliaferry et al. [11] explored computational methods for accounting for the risk attitude of the skipper, adopting the discrete DP-based solution earlier proposed in [1] for route planning under uncertainty.

In summary, researchers working on optimal route planning

for sailcraft have already been experimenting with MDP formulations of the problem, but due to the limitations of the previously dominant value-function-based approach to solving MDPs (and other representations of RL problems), these attempts have resorted to rather coarse discretizations of the state and action spaces of the problem, leading to fairly significant and probably quite sub-optimal approximations and simplifications. In the meantime, after recognizing the need to solve planning problems in physical systems with naturally continuous state and action spaces, research in the field of RL has also explored an alternative solution approach based on policy gradients and optimization directly in policy space that can handle continuous states and actions more easily [12]. Combined with the unparalleled function approximation power of deep neural networks, this approach has led to the successful solution of a number of benchmark control problems of significant complexity [13]. This suggests that such algorithms might bear fruit on the problem of optimal navigation of sailcraft under wind uncertainty, which is the main idea investigated in this paper.

III. Equations of Motion of a Boat Sailing Upwind

Any sailboat is a complex vessel with many moving parts, each of which has its own dynamics of motion that interact with the force and direction of the wind to produce motion. Consequently, the dynamics of a sailboat can be modeled at various levels of detail, depending on objectives. For our purposes (computation of optimal navigation), a relatively simple kinematic model will suffice, while still matching well trajectories taken by real boats during races. In this model, the boat is assumed to be a point with position coordinates (x, y) and direction (heading) θ, controlled by means of a rudder that can adjust the heading by an amount u every control step, $u_{min} \leq u \leq u_{max}$, with $u_{min} = -u_{max}$. If the control time step is of duration Δt, this is equivalent to setting the rotational velocity of the boat to $u/\Delta t$ during a control step.

If x_k, y_k, θ_k, and u_k are the position, direction, and control variables at time $t_k \doteq k\Delta t$, the state evolution of the boat during discrete control steps $k = 1, K$ can be obtained by approximating its continuous-time motion by means of forward Euler integration, to obtain the discrete-time equations

$$\begin{aligned} x_{k+1} &= x_k + \cos\theta v_k \Delta t \\ y_{k+1} &= y_k + \sin\theta v_k \Delta t \\ \theta_{k+1} &= \theta_k + u_k \end{aligned} \tag{1}$$

These equations of motion distill the complicated interplay of all forces acting upon the hull and sails of the boat, including effects such as heeling (leaning) of the boat sideways in response to these forces, into the single variable v_k that describes the surface speed of the boat during control period k. The velocity $v_k = v_k(\alpha_k, h_k)$ is a function of the difference $\alpha_k \doteq \theta_k - \omega_k$ between the heading of the boat θ_k and the current true wind direction (TWD) ω_k, as well as the current true wind speed (TWS) h_k, and can be obtained directly from the polar diagram of the boat. The direction of the wind ω_k is as it would be experienced by a stationary observer not in the moving boat.

IV. Reinforcement Learning Problem Formulation

The objective of this paper is to formulate the problem of optimal sailing upwind as a reinforcement learning problem, and solve it by means of advanced RL algorithms. Since the underlying mathematical formalism of most RL algorithms is that of Markov decision processes (MDP), our problem needs to be represented as an MDP [10]. As mentioned above, the use of MDPs in optimal route planning for sailboats has been proposed in the past [2], and our work builds upon this idea, expanding the formulation into a form suitable for solution by means of modern RL algorithms [4].

An MDP is a discrete-time stochastic control process that is described by the tuple (S, A, P_a, R_a), where S is the state space, A is the action space, $P_a(s, s') = Pr(s_{k+1} = s' | s_k = s, a_k = a)$ is the probability that taking action a in state s at decision step k will lead to state s' at the next time $k + 1$, and $P_a(s, s')$ is the immediate reward, respectively cost, resulting from transitioning from state s to state s' due to choosing action a [10]. The objective of the reinforcement learning algorithm is to find an optimal policy π that selects actions $a_k = \pi(s_k)$ that maximize some cumulative measure of reward, such as the expected discounted future reward for tasks with infinite duration, or the average (or total) reward per episode for episodic tasks of finite duration.

A. Continuous-Action MDP

The MDP in a RL problem includes the entire environment that affects how rewards will be obtained as a result of action choices. The sailboat, with its equations of motion (1), is one component of the environment, represented by the three variables x, y, and θ. Note than unlike in previous work (e.g. [2]), where these state variables were discretized, in our formulation they remain continuous, and we will rely on the power of deep neural networks to represent policies and value functions over them. In addition to these three state variables, another necessary component of the environment, respectively the MDP modeling it, is the wind, whose angle TWD and speed TWS also need to be included in the environment. Unlike the equations of motion of the sailboat, which are assumed to be deterministic and are already in the format needed for inclusion in the MDP, the dynamics of the wind are generally stochastic, and a suitable model is necessary.

Wind modeling is an active area of research in its own right, and for the purposes of this study, we need a wind model that is reasonably detailed in order to capture the variability typical during sailboat races, but is still compatible with the MDP formalism. One such model is the Ornstein-Uhlenbeck (OU) process, a stationary Gauss-Markov process that has mean-reverting properties, that is, over time it tends to move towards a central value [14]. This property matches quite well the characteristics of the wind over the relatively short time of a single leg in a boat race: the wind has a dominant direction, along which the windward mark and the starting

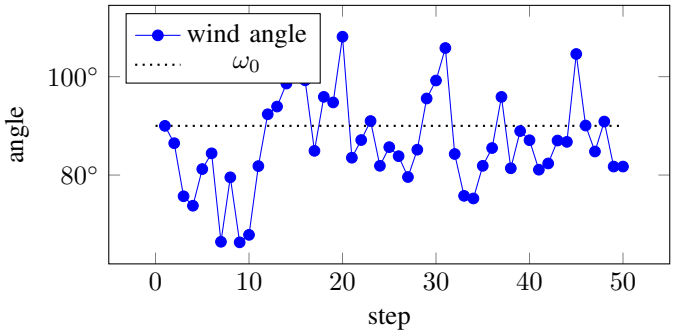

Fig. 3. A realization of the stochastic process for wind direction.

position of the boat are located, but can often vary quite significantly from this dominant direction over the duration of the race, eventually and occasionally returning to it. This return tends to be relatively gradual, because wind direction tends to be determined by relatively large air masses that do not change their direction very abruptly. The same argument can be made for the wind velocity, too. This kind of mean-reverting stochastic processes for the wind have already been used in optimal sailboat planning [1].

Furthermore, the stationarity property of an OU process is essential in an MDP, where it is assumed that the transition probability function $P_a(s, s')$ is constant over time. Moreover, this process is first-order Markovian, making it particularly easy to integrate into the transition function of an MDP. In discrete time, it can be represented by an autoregressive process of order 1 ($AR(1)$). For example, the wind direction can be expressed by the equation

$$\omega_{k+1} = c + \phi\omega_k + \epsilon_k,$$

where c and ϕ are suitably chosen constants, and ϵ_k is a random normally distributed variable that is uncorrelated with any previous random values ϵ_{k-1}, ϵ_{k-2}, etc. The mean value μ that ω tends to revert to will be $c/(1-\phi)$. We will also make the assumption that the initial direction of the wind $\omega_0 = \mu$, as this is a condition that officials in sailboat races can ensure by adjusting the starting position of the boats accordingly. An example of a realization of the wind direction process is shown in Fig.3. In this graph, ω_0 is set to 90°, also $c = 0.3\omega_0$, $\phi = 1 - 0.3 = 0.7$ in order to ensure that μ will be equal to the initial value ω_0. As for ϵ_k, its standard deviation was set to 10° and its mean was 0°.

If we combine the three state variables of the sailboat, x, y, and θ, with the two state variables of the stochastic process of the wind, ω and h, we will obtain a five-dimensional continuous state space of a resulting MDP whose action set is described by the single continuous variable u that expresses the correction to the boat's heading within a control step. The reward function for this MDP could be defined as follows: for every control step during which the boat has not reached its goal state yet, it is given a reward of -1 units, with the meaning of time cost. The goal is reached when the boat crosses the line from the windward mark directly in windward direction, in the clockwise direction (shown in a dashed line in Fig. 1). (Although in races the finish line is usually oriented perpendicular to the wind, we have chosen the finish line in this problem to be parallel to the wind direction, because the objective of the boat here is not only to reach the windward mark, but also to round it.) If the boat crosses the finish line, the episode terminates. If the boat cannot reach the goal within K time steps, for some sufficiently high value of K, the episode terminates and the cumulative reward is set to $-K$. Thus, essentially, we are treating this problem as episodic RL, and the negative cumulative reward of an episode is equal to the time it would take the boat to reach the goal line.

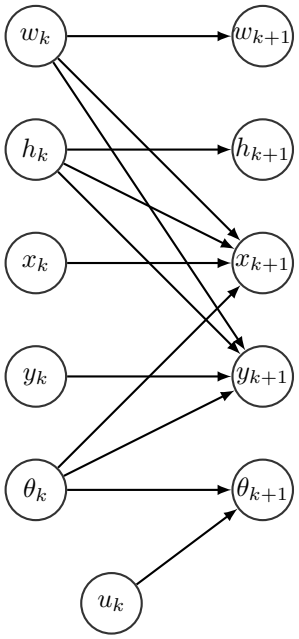

Fig. 4. Environment dynamics represented as a dynamic Bayesian network. Nodes represent variables, and edges represent dependencies between variables. The lack of an edge between two variables indicates conditional independence between them. The control variable u_k for a particular period k depends on all state variables at that time through the chosen control law (now shown in the figure).

The dynamics of the environment of the MDP defined in this manner can be represented by the dynamic Bayesian network (DBN) shown in Fig. 4. The state of a DBN is represented by a collection of random variables, and two consecutive states s_k and s_{k+1} are shown along with the effect each of these components of the state vector has on all components at the next time step [15]. An edge between two variables shows conditional dependency, and the lack of an edge shows independence. Note that, in principle, higher-order dependencies can be expressed by introducing edges from time steps before the immediately preceding one. For example, an autoregressive process $AR(p)$ of arbitrary order p for the wind can be expressed this way. What is not shown in this diagram is the dependency of the control variable u on all components of the state vector; it is the job of the boat controller to decide what this dependency should be, in the form of a control law.

B. Discrete-Action MDP

Although the MDP described above is of the form that modern RL algorithms can attempt to solve, knowledge of maritime navigation accumulated over the centuries, as well as previous research in the area highlighted above, suggest an additional MDP formulation based on the way human skippers sail upwind that can reduce the state space of the problem and simplify its decision variable. The idea is to restrict the action space of the problem to only the choice of tack on which the boat will be sailing upwind, similar to the approach taken by Ferguson and Elinas [2]. This is equivalent to making a high-level decision only about the sailboat's tack (port or starboard), and then assuming a default point of sail for that tack determined by the intermediate-level helming controller.

As noted above, when sailing upwind, the usual practice is to always keep the boat heading at the optimal angle α^* with respect to the TWD that maximizes the VMG, and thus progress towards the goal. That is, if the TWD at time period t_k is w_k and the TWS is h_k, the helming controller will first need to find the optimal angle $\alpha_k^* = \text{argmax}_\alpha[v(\alpha, h_k)\cos\alpha]$, where $v(\alpha, h)$ is the velocity for TWA α and TWS h as represented by the polar diagram of the boat. After that, if the decision is to sail on starboard tack, the optimal heading of the boat is computed as $\theta_k^* = \omega_k + \alpha_k^*$, and if the decision is to sail on a port tack, the optimal heading of the boat is chosen to be $\theta_k^* = \omega_k - \alpha_k^*$. After that, the desired optimal heading θ_k^* is given to the low-level steering controller, which will take corrective action to bring the current heading θ_k to the desired heading θ_k^*. One possible choice for the steering controller is to simply command the change in direction to be $u_k = \theta_k^* - \theta_k$, if $-u_{max} \leq \theta_k^* - \theta_k \leq u_{max}$, or the respective extreme changes, $-u_{max}$ or u_{max}, otherwise.

Note that although we are using the same two actions as in [2], our discrete-action MDP is not the same as theirs. The first big difference is that whereas their Markov model had a discrete state space, following an explicit discretizaton of the position and wind variables, our MDP has a completely continuous state space, and only its actions are discrete. Second, it does include the boat's heading as a state variable. Moreover, whereas their Markov model performed a tacking maneuver in one time step, and required an extra time penalty to be assessed for tacking, in our case a tacking maneuver naturally takes as many steps as it needs for the low-level steering controller to bring the boat from its current heading θ_k to the desired heading θ_k^*, subject to the limitations of the rudder angle. When changing tacks, this would be a relatively large change, on the order of $2\alpha_k^*$, that might take several time steps. The motion of the boat at that time will slow down accordingly, so by simply assessing a reward (negative cost) of -1 for each time step, as in the continuous-action MDP, the slowdown due to tacking will be correctly accounted for. Another benefit of using our formulation is that the trajectory of the boat will curve naturally, similar to that of real boats, as opposed to doing sharp turns in place that are not realistic.

V. Empirical Verification

A. Experimental Set-up

The MDPs described in the previous section, which represent the sequential decision making problem of route selection when sailing upwind, were encoded as two OpenAI Gym environments [16], one with discrete and the other one with continuous actions. Leveraging the standard format of these two OpenAI Gym environments, several algorithms from the RLlib library for reinforcement learning were used on them: Proximal Policy Optimization (PPO, [17]), Soft Actor Critic (SAC, [18]) and Augmented Random Search (ARS, [19]). PPO is an algorithm that updates its network parameters in certain range by restricting the amount of update. SAC is a modification to the Deep Deterministic Policy Gradient algorithm (DDPG, [13]). SAC provides better exploratory learning compared to DDPG by including an entropy term in its objective function. ARS is a model-free algorithm and one of random search algorithms which achieved good performance with discrete actions.

In the simulation environment, the same course was raced again and again for varying wind patterns generated from the same stochastic process that is unknown to the learning algorithm. The goal location was located 5 nautical miles directly upwind from the starting location. Both locations were kept constant. The time step of the simulation was set to 1 minute. The polar diagram used in simulation was that of the Bavaria Cruiser 32 sailboat, and the wind speed was assumed to be constant at 10 knots. (Variations in wind speed affect the speed of the boat, but their impact on sailing strategy is not so large, so in this study, we ignored them.) The standard deviation of the random deviations in the AR(1) process of the wind was 10°, and $c = 0.3\omega_0$, $\phi = 1 - 0.3 = 0.7$.

During learning, each RL algorithm repeatedly sampled the Gym environment over and over again, each time testing its control policy over a new realization of the wind process. However, once training has finished, for the sake of fair comparison between all controllers, we evaluated all of them on the exact same $1,000$ wind patterns, and report their average cumulative rewards (respectively, their average time to reach the goal).

B. Certainty Equivalence Model Predictive Controller

As a baseline for comparison, we implemented two basic model predictive controllers that ignore the variability of the wind, and always plan the future course of the boat as if the current direction of the wind will persist until the end of the course. For this reason, we call them certainty equivalence MPC controllers. They can still reach the mark successfully because, following the principle of model predictive control (MPC), at each step, they re-plan the control and state sequence to the mark from the boat's current position, take the first action, and if the next position does not match its original expectation, they still get to re-plan from the actual next position on the next control step.

The two MPC controllers use the same logic about when to change tack, and the only difference between them is how

they choose the boat heading when on starboard tack. The tack-selection logic follows the general strategy when sailing upwind that was already described above. If the wind will remain constant in speed and direction, the boat can reach the mark in minimal time by first sailing on a port tack until it reaches the layline, then tacking to starboard, and then sailing on starboard tack. Of course, while sailing on starboard, the wind might shift, and the boat might find itself on a course that will no longer round the mark on the right side. In that case, it will need to tack back to port, and sail on port tack until it crosses the layline again, so that it can go back on starboard tack.

This suggests the following rules for deciding when to change tack or stay on the current one. First, the current angle to the mark is computed as $\beta_k = \arctan[(y^{(m)} - y_k)/(x^{(m)} - x_k)]$, where (x_k, y_k) is the current position of the boat, and $(x^{(m)}, y^{(m)})$ is the position of the mark. Then, when on port tack, the boat should tack to starboard if $\beta \geq \omega_k + \alpha_k^*$, that is, if it would *fetch* (manage to round) the mark on starboard tack from the current position. If it cannot fetch the mark, the boat should stay on port. Conversely, when the boat is sailing on a starboard tack, if the same condition $\beta \geq \omega_k + \alpha_k^*$ is true, the boat can stay on starboard, expecting to fetch the mark, and tack to port, if the condition is not true.

Both versions of the MPC controller use the same logic for changing tack described above, and also both of them always steer on course $\theta_k^* = \omega_k - \alpha_k^*$ when on port tack, as it is the course with the greatest VMG, that is, making the most progress upwind. Where the two MPC controllers differ is in the chosen heading when on starboard tack. One of the controllers chooses the symmetric heading $\theta_k^* = \omega_k + \alpha_k^*$, on the logic that it, too, will make the most progress upwind. We call this controller MPCd (for discrete), as its behavior matches that of the discrete MDP described above: it only chooses the tack, and the heading is chosen for it automatically by the helming controller in the MDP.

The other MPC controller recognizes the fact that the boat would be sailing on starboard tack only if it can fetch the mark from the current position with the current wind direction ($\beta \geq \omega_k + \alpha_k^*$), so it can succeed in rounding the mark on any heading in the interval $[\omega_k + \alpha_k^*, \beta]$. Choosing the lower end of the interval, $\omega_k + \alpha_k^*$, as the MPCd controller does, is probably not going to be optimal, because the boat will sail at a lower speed, and travel a longer distance to the finish line (the orange trajectory in Fig. 1) . If the wind direction will remain constant, as the certainty-equivalent controller believes it will, the optimal choice will be to head directly for the goal, that is, select a target heading equal to β. The command u_k to the low-level controller can be computed then as $u_k = \beta - \theta_k$, subject to its lower and upper limits. The resulting trajectory, for constant wind direction after the tack to starboard, is shown in green in Fig. 1.

We will call this controller MPCc (for continuous), because it computes directly the continuous control signal u, as opposed to only the desired tack, as MPCd does. It will be our primary choice for a baseline controller for comparison, because its operation matches closely how the vast majority of sailors would sail upwind, particularly in non-competitive situations. However, even though the MPCc controller is likely to be more optimal than the MPCd controller, it is easy to see why it would not be completely optimal. When sailing on starboard tack, going straight for the mark is somewhat risky, because the boat would sacrifice progress against the wind for faster speed. Later, an unfavorable change of wind direction towards the bow of the boat (known as a *header*), might make it no longer possible to fetch the mark on the current starboard tack, and the boat might have to tack to port and then later back to starboard, thus losing time. So, it might be advantageous to sail on a course somewhat higher upwind, but it is not clear what the optimal angle might be. Moreover, this angle would depend on the boat's current position and the difference between the current wind direction and its predominant direction. Choosing this angle optimally could be done by means of a more advanced sailing strategy, of the kind an RL algorithm might learn. Another source of suboptimality for both the MPCd and MPCc controllers is the decision logic about when to tack. When sailing on port tack, if the current direction of the wind has resulted from a header, waiting to tack to starboard on the current layline might be too late; if the wind returns to its predominant direction, the boat would end up too far upwind. (That is one way for the boat to find itself on the green trajectory in Fig. 1.) In this situation, it might be advantageous to tack to starboard early, before crossing the current layline. This, however, is risky, too, in case the wind remains unfavorable. (That is how the boat might end up on the red trajectory in Fig. 1.) This reasoning illustrates how the tacking logic of a more advanced sailing strategy might improve upon that of the MPC controllers. Nevertheless, with the understanding that the MPC controllers might be somewhat suboptimal in comparison to what an experienced skipper might do in a race, they make for reasonable baselines for comparison.

TABLE I
TRAINING TIME OF RL ALGORITHMS.

	Discrete		Continuous	
	PPO	ARS	PPO	SAC
hours	26.0	6.6	20.6	17.7

C. Reinforcement Learning Policies

The learning algorithms were run on a computer with a 12-core CPU, using PyTorch and RLlib. Table I shows how many hours it took to compute these policies for each algorithm.

D. Empirical Results

After the RL agents completed training, we evaluated each of the learned controllers along with the two MPC controllers over the exact same $1,000$ realizations of wind direction. Table II shows the average and standard deviation of the negative cumulative reward, corresponding to sailing time from start to finish. It is evident that all deep RL algorithms outperformed

TABLE II
AVERAGE DISTANCE TO GOAL FOR ALL CONTROLLERS, IN MINUTES, OVER THE SAME 1,000 WIND REALIZATIONS.

	Deep Learning				MPC	
	Discrete		Continuous		Disc	Cont
	PPO	ARS	PPO	SAC	-	-
mean	85.2	85.4	**83.2**	83.5	92.9	90.25
stdev	6.7	19.1	10.5	6.1	17.0	9.1

TABLE III
WINNING PERCENTAGE OF THE RL CONTROLLERS AGAINST MPCC.

	Discrete		Continuous	
	PPO	ARS	PPO	SAC
win rate	80.3%	75.9%	**87.4%**	82.9%

significantly both MPC algorithms. The shortest sailing time was achieved by the continuous RL controllers trained with the PPO and SAC algorithms, which shorten sailing time by 7 minutes on a race course where the MPC algorithms took more than an hour an a half (90 minutes) to complete. A difference of this magnitude would be overwhelming in a real sailing race.

As the standard deviations of sailing times for all controllers are rather large, due to the high variability of the wind patterns encountered during testing, it is not very meaningful to do unpaired statistical testing between sailing times of two controllers by using an unpaired test using only the population means and standard deviations of the two controller's sailing times. Instead, because testing was done on the exact same $1,000$ wind patterns, it is more informative to do a paired statistical test. In fact, in this domain, an obvious such test exists – how often would each of the two controllers win the race, if sailing in the exact same wind conditions. Table III shows the percentage of the $1,000$ races that each RL algorithm won against the better of the two MPC controllers, MPCc. As expected, the advantage in average sailing time translates to a high chance of winning: the best of the RL controllers, PPO with continuous actions, wins against MPCc in 87.4% of the races.

It is also interesting to observe the progress of each RL algorithm across time. Figs. 5 and 6 show how each RL algorithm learns to improve the cumulative reward per episode. For both figures, the performance of the respective baseline MPC controller is shown for comparison, along with the uncertainty band implied by its standard deviation in sailing times. Fig. 5 shows that although both PPO and ARS with discrete actions (tack decision only) eventually achieve a comfortable lead of around 5 minutes over MPCd, PPO learned much faster, learning how to sail well in around 10^4 episodes. A similar comparison can be made between PPO and SAC with continuous actions: both reach approximately the same asymptotic performance, but PPO needs a lot fewer episodes to learn. Its behavior is also a lot more purposeful, starting to improve performance almost immediately, whereas SAC fails

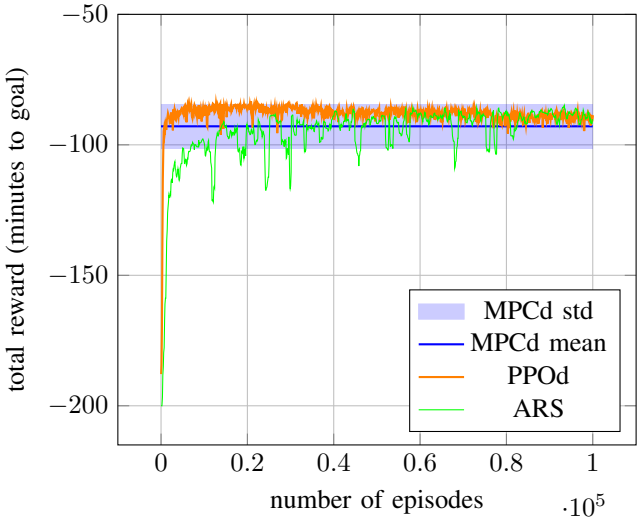

Fig. 5. Cumulative reward in discrete action space. PPO learns much faster.

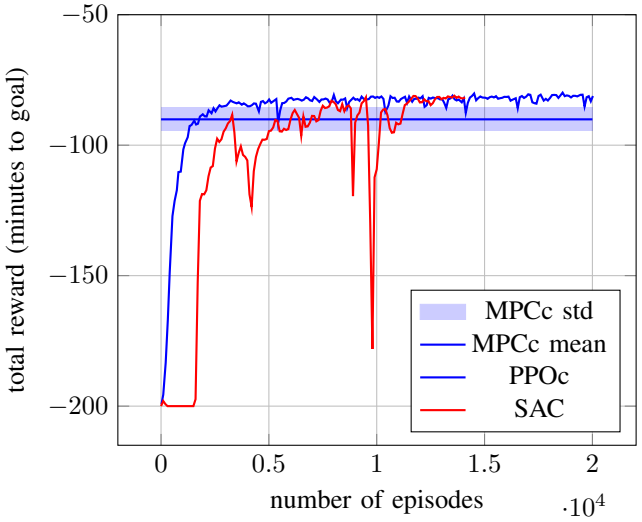

Fig. 6. Cumulative reward in continuous action space. PPO starts learning immediately and reaches good performance much faster.

to even round the mark once for the first $2,000$ episodes. Based on these results, the clear winner is the PPO algorithm, for both MDP formulations of the problem.

And, finally, it is instructive to look at the actual routes all controllers selected, and try to distill the general strategy they are using. Figs. 7 to 10 illustrate the routes of sailboats under four patterns of wind direction selected to represent a usual pattern (Fig. 7), a favorable pattern where the wind shifts to the east and thus helps progress to goal (Fig. 8), an unfavorable pattern where the wind shifts to the west and thus hinders progress (Fig. 9), and finally an unusual pattern where the wind fluctuates quite wildly (Fig. 10). In Fig. 7, MPCs tacks early, and then its tacking logic forces it to tack two more times. The other controllers tack later, and never have to tack again. This kind of decision, to tack a bit after the layline is crossed

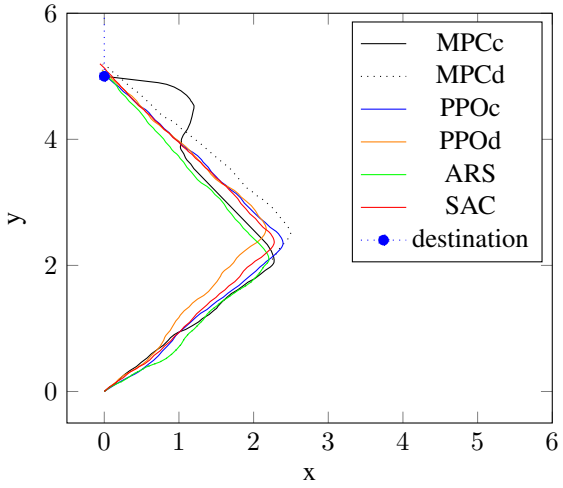

Fig. 7. A usual wind direction pattern.

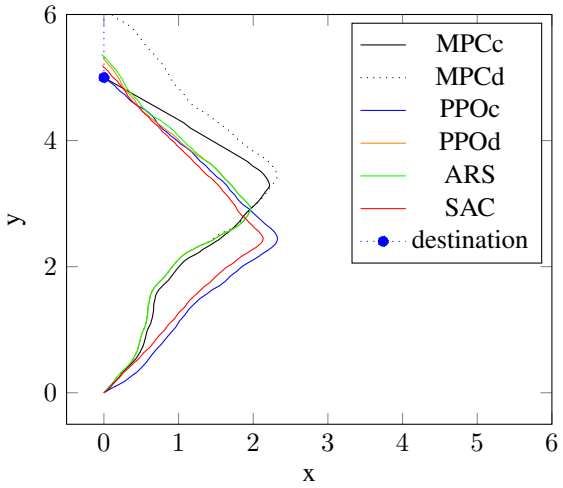

Fig. 8. A favorable wind direction pattern.

is call *overstanding* in sailing, and knowing when to overstand is an essential element of competitive racing. Interestingly, the ARS controller also tacks as early as MPCc, but decided to stay the course. This kind of approach to the goal at an angle to the wind less than the optimal angle α^*, in the hope of rounding the mark without having to tack, is called *pinching*, and is also a part of competitive strategy. A later favorable shift of the wind probably helped, too.

Similar early tacking by MPCc is seen in other wind patterns, too – in fact, it tacks too early in three out of the four patterns, and is then forced to tack two more times, clearly indicating that this is a major reason for its suboptimality. Based on this discovery, it is tempting to try to tune MPCc to overstand by some constant amount, but experiments with this strategy did not improve its average sailing time, as the optimal amount of overstanding is position- and wind condition-dependent. In contrast, the deep RL controllers were able to learn the correct action as a function of the entire state space, giving them an advantage.

VI. Conclusions and Future Work

In this paper, we described the application of deep RL algorithms to the problem of learning to sail upwind optimally. The main contribution of the paper is the formulation of the sailing upwind problem under wind uncertainty in the form of a Markov decision process that would allow modern deep RL algorithms to be applied to it. In an empirical verification, all three RL algorithms we experimented with (PPO, SAC, and ARS) were able to reliably achieve a significant advantage over baseline MPC algorithms that closely mimic how novice sailors sail upwind. In an effort to make the problem easier for RL algorithms to solve, we provided two MDP formulations, one of which had a limited set of actions specifying only which tack the sailboat should be on, such that each of the two actions reliably moved the boat towards the goal, as opposed to wandering aimlessly around the ocean. It turned out that this kind of simplification and assistance was not

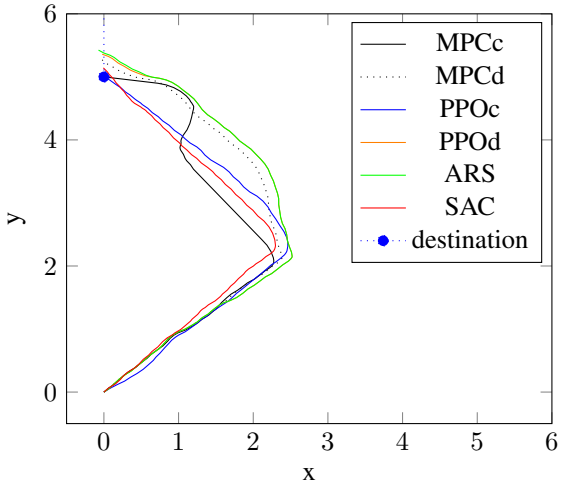

Fig. 9. An unfavorable wind direction pattern.

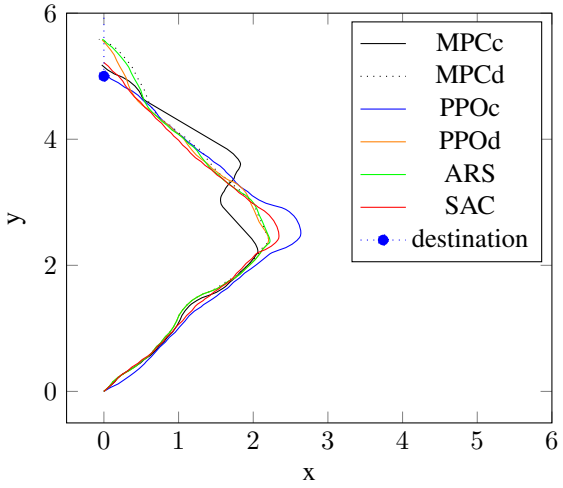

Fig. 10. An unusual wind direction pattern where the wind fluctuates a lot.

even necessary – the continuous-action MDP formulation that used directly the rudder correction as its action was not only solvable by the PPO and SAC algorithms, but the resulting controllers achieved the best performance overall. The controller computed by the PPO algorithm was able to decisively outperform the MPC controller, winning 87.4% of all races against it, which clearly demonstrates the potential of deep RL algorithms in this class of sequential decision making problems under significant uncertainty. This opens the way for application of deep RL technology to robotic sailcraft, including fleets of saildrones that can explore the oceans of the planet autonomously for indefinitely long periods, using only wind and solar power.

However, this kind of superior performance comes at a rather high computational price at the moment. The best overall algorithm, PPO, took over 20 hours to compute the optimal policy. Although the resulting policy is an entire control law that can be computed once and uploaded to a sailcraft for use, the wind process will likely change before the computation is over. However, by using more advanced RL algorithms and faster computers, a reduction in computation of several orders of magnitude might be possible in the near future. Furthermore, related problems involving other types of vessels with more stable (and still stochastic) operating conditions might benefit from a similar type of decision making technology.

REFERENCES

[1] A. Philpott and A. Mason, "Optimising yacht routes under uncertainty," in *15th Chesapeake Sailing Yacht Symposium*. OnePetro, 2001.

[2] D. S. Ferguson and P. Elinas, "A Markov decision process model for strategic decision making in sailboat racing," in *Canadian Conference on Artificial Intelligence*. Springer, 2011, pp. 110–121.

[3] R. Ferretti and A. Festa, "A Hybrid control approach to the route planning problem for sailing boats," *arXiv preprint arXiv:1707.08103*, 2017.

[4] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[5] D. H. dos Santos and L. M. G. Goncalves, "A gain-scheduling control strategy and short-term path optimization with genetic algorithm for autonomous navigation of a sailboat robot," *International Journal of Advanced Robotic Systems*, vol. 16, no. 1, p. 1729881418821830, 2019.

[6] S. Lemaire, Y. Cao, T. Kluyver, D. Hausner, C. Vasilovici, Z.-y. Lee, U. J. Varbaro, and S. M. Schillai, "Adaptive Probabilistic Tack Manoeuvre Decision for Sailing Vessels," in *International Robotic Sailing Conference*, 2018, pp. 95–103.

[7] F. Plumet, C. Petres, M.-A. Romero-Ramirez, B. Gas, and S.-H. Ieng, "Toward an autonomous sailing boat," *IEEE Journal of Oceanic Engineering*, vol. 40, no. 2, pp. 397–407, 2014.

[8] J. E. Kerwin and J. N. Newman, "A summary of the H. Irving Pratt ocean race handicapping project," in *4th Chesapeake Sailing Yacht Symposium*. OnePetro, 1979.

[9] R. Stelzer and T. Pröll, "Autonomous sailboat navigation for short course racing," *Robotics and autonomous systems*, vol. 56, no. 7, pp. 604–614, 2008.

[10] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[11] F. Tagliaferri, A. B. Philpott, I. M. Viola, and R. G. J. Flay, "On risk attitude and optimal yacht racing tactics," *Ocean Engineering*, vol. 90, pp. 149–154, 2014.

[12] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[14] A. Neumaier and T. Schneider, "Multivariate autoregressive and Ornstein-Uhlenbeck processes: estimates for order, parameters, spectral information, and confidence regions," *ACM Transactions in Mathematical Software*, 1998.

[15] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[16] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[19] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," *arXiv preprint arXiv:1803.07055*, 2018.