Short-term Path Planning for High-level Navigation Control of N-Boat - The Sailboat Robot*

Davi Santos[1], João Villas Boas[1], Alvaro Negreiros[1], Andouglas Junior [1], Justo Alvarez[1], and Luiz Gonçalves[1]

Abstract— **The use of wind-propelled boats has becoming an interesting approach to ocean missions that require long duration tasks and both low-cost and green computing characteristics. To exemplify the looking in this direction, some initiatives have started as competitions to traverse the Oceans, as the *Microtransat Challenge*. Some issues have to be worked on, at the research level, for allowing a completely autonomous sailboat robot operation such as short-term path planning, sailboat control, sensors data collection, and electrical supply (mainly based on green computing models). Particularly, the short path planning of a robotic sailboat is a specially challenging task, once the sailboat can not reach points that are directly against the wind following a straight path given by some higher-level trajectory planner. Towards solving this issue, this work proposes a short-term path planning method for autonomous sailboat that is capable of dealing with upwind situations. In order to accomplish this, an initial path is geometrically defined and an optimization is done over this path, using genetic algorithm. Basically, our path planning method uses the distance ranges available (longitudinal and lateral) for the maneuvering and the sailboat desired heading to generate points in-between, which are reachable given the wind restriction. The optimization method based on GA uses these path planning parameters to come up with finding an optimal path. The results demonstrate feasibility and usefulness of the method.**

I. Introduction

Over the past 15 years, the number of researches on autonomous sailboat has grown rapidly. The main motivation for building robotic sailboats is the possibility of performing long endurance autonomous missions (weeks or months), without human intervention, using both low-cost and green computing solutions. This is possible because, while conventional robotic boats spend most of their energy with propulsion (no matter if using fuel or electricity powered), sailboats are wind-propelled, however, only for given direction. This feature can be useful in long term missions and environmental such as coral reef monitoring, water quality monitoring and border surveillance.

There are several challenges related to autonomous robotic sailing. The embarked electronic equipments in the sailboat should be inside properly sealed compartments. Redundant sensors must be added to the system in order to maintain its autonomy in a robust way. Also, as the sailboat works in a highly variable environment with sudden changes in wind velocity and direction, control laws must be efficient, effective and robust. Sailboat path planning is also a challenging task, due to the existence of a region against the wind where the sailboat cannot gain speed, called dead area. In these situations an expert (human) sailor would perform an step pathway (a zigzag) to reach the desired point. This maneuver is usually called beating or tacking, using sailing terminology.

The literature presents a few researches on upwind path planning for sailboats and its optimization. Basically, there are two types of approaches to the problem: analytic and probabilistic. The analytical approach takes into account local and instantaneous information of the wind velocity and direction to find the path, while the probabilistic approach finds the optimal path using information about the probability distribution of the wind.

In the analytical approach, strategies such as potential fields, representing the dead area as an obstacle [1], observation of the optimal heading given by the sail polar diagram [2] and others that use fixed values as heading [3] [4] can be found in the literature. All of these aforementioned works are implemented and validated with real experiments using a robotic sailboat. Strategies based in the probabilistic approach, such as Markov Decision Processes [5], [6] that finds the best policy through simulations and Markov Chains [7] that presents an estimation of the best path, are even becoming to be used in sailboat regattas where the boats are operated by humans. It is worst to mention that none of the strategies that presents a probabilistic approach for the problem has yet been implemented in a robot sailboat.

Is this work, we have developed a path planning technique that allows a robotic sailboat to autonomously perform a beating maneuvering. Basically, this method uses information of the available area for the maneuver and the sailboats desired heading to find reachable way-points in-between. We first developed the mathematical model of the sailboat and method itself and then for generating trajectory in upwind situations, later implemented and tested in simulation. Furthermore, we have developed and performed a model based optimization of the path planning method, using genetic algorithms (GA). The GA uses the path planning method parameters to generate multiple trajectories, finding the optimal path through evolutionary processes. The method is implemented and tested using a simulator.

We remark that a first control of the sailboat is tested in practice in a sailboat model, the N-Boat I, which is an RC Monsoon 900. This model uses servos as actuators for the sail and rudder control. In this case, a simple proportional controller was used [9] and showed to work in practice [10]. However, situations against wind were not tested at this first implementation. So, in the second prototype that is being finalized now (Figure 1), a more complete algorithm has been devised and is introduced in this work. So the main contribution of this work is the short term path planning

including situations of upwind.

Fig. 1. N-Boat II Robot measuring 2.5 meters in lenght

II. The Theory of Sailing

Sailing is known to be one of the first approaches developed by humans in order to carry cargo by long distances [10]. Basically, two system components are essential to control the sailboat movement: the rudder, that control the course changes, and the sail, that uses the wind to control the boat's velocity. To reach a desired target, sailors use their experience to carefully steer the rudder and adjust the sail angle. Usually, cables for the sail and a tiller for the rudder are generally used for manually controlling a sailboat, which should be automated in a robotic sailboat.

A. Sailing Manouvers

It is worth noting that not all target directions can be reached by the sailboat. In fact, given the direction of the wind, it is impossible to get the boat to move towards some angular region against the wind, as shown in Figure 3, say some 80 to 100 degrees wide depending on the sailboat design. The boat cannot gain velocity when pointing towards this conic region called the dead area. If a sailboat stays too long pointing to the dead area it looses velocity and will eventually stop. After all, changes in the rudder will no longer have effect, and the sailboat is said to be out of control. This situation should be avoided at all costs.

TABLE I
Adopted Notation

Notation	Description
a_X, b_X	Slope and y-intercept of line X
P_0, P_d	Initial and target points
θ_t, d_t	Tacking angle and distance
$\theta_{d_W}, \theta_{v_W}$	Heading to target and wind direction on the W frame
θ_{v_B}	Wind direction on the B frame
t_a	Arrival time in seconds
$Proj_{m,A}$	Projection of point m in line A
$d_{a,b}$	Euclidean distance between points a and b
N_{ger}	Number of generation

To follow a desired heading, one needs to carefully adjust the rudder and sail positions. There are, basically, the following three situations of motion with a few variation, that should be treated separately:

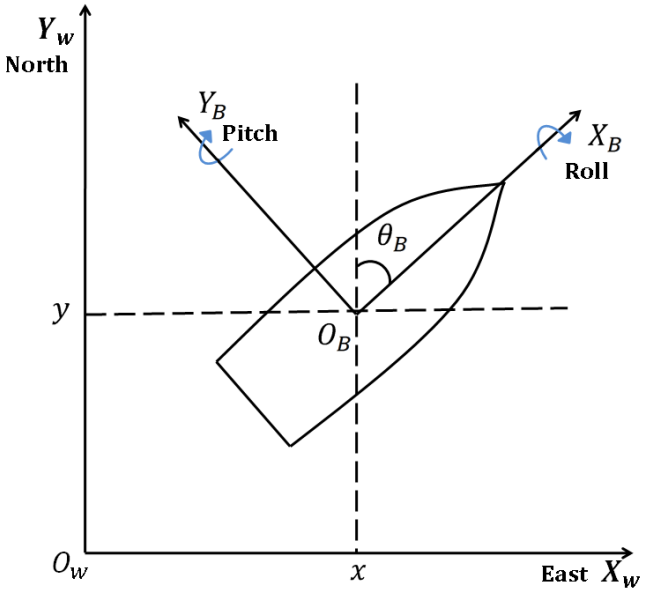

Fig. 2. General coordinate system for a sailboat

Towards the wind direction
Formed by the Running and Broad Reaching maneuvers. In this situation, the sailor release some of the sail's cable to gather the maximum possible wind and get velocity.

Wind to port or to starboard
The wind is lateral to the boat, so to get velocity, the sailor just need to release about half of the cable size in such a way that the sail stays in about a quarter of the circumference (45 degrees) direction with respect to the wind direction.

Against the wind direction
It is not possible for the sailboat to gain velocity in this situation, however if the target is in this closed hauled region it can be reached using some strategy. If the destiny is directly against the wind the sailor just has to perform a zigzag motion alternating in lines on either sides of the wind direction (tack). This process is called tacking (or beating).

The third situation above suggests that another path has to be calculated and optimized in order for the boat to reach the goal. We have adopted in this work the use of genetic algorithms as a way for determining and optimizing a final path for the boat to tack over it and reach the goal when it is against wind direction.

B. Genetic algorithm

Genetic Algorithm (GA) is a search method, usually applied in learning and optimization, which is based on the adaptive behavior concepts formalized by Darwin in his Theory of Natural Evolution. GA are a type of non-deterministic algorithms, i.e., each run may lead to different

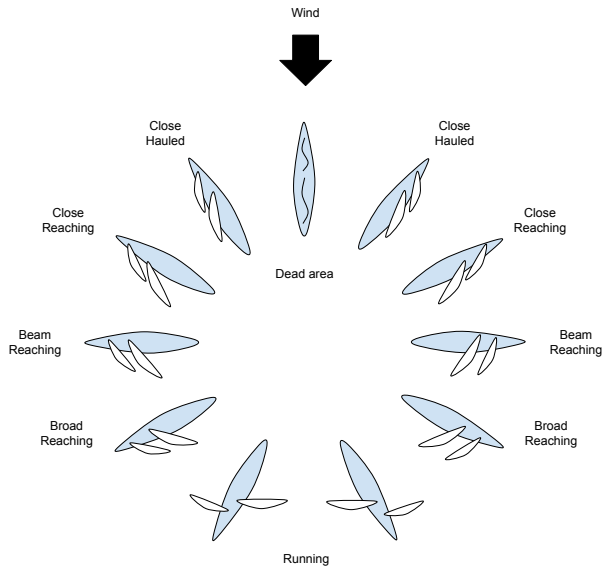

Fig. 3. Possible maneuvers of sailing

results. The classical implementation of a GA is shown in figure 4. Initially, a random population is created, where each individual represents a possible solution to the problem. Then, individuals are evaluated using a fitness function, that indicate how good each individual is.

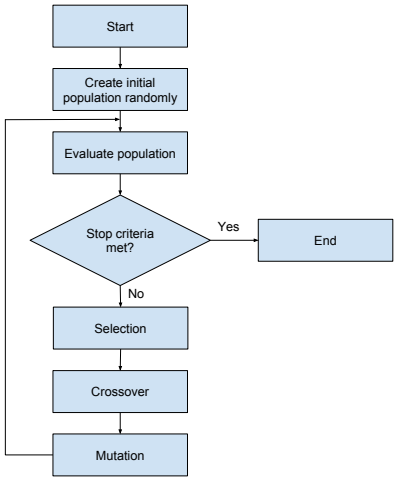

Fig. 4. Block diagram showing the general vision of the GA working.

A selection is made where individuals with higher fitness are more likely to be chosen, while the remaining individuals are discarded. The selected individuals are then modified through genetic recombination and mutation, creating the individuals of the next generation. This process is repeated until a satisfactory solution is obtained, or some other stopping criteria is met. Each iteration of this process is called a generation. It is worst to mention that GA is a well defined tool for optimization and machine learning [11].

C. The N-Boat Simulator

Our method is tested using a sailboat simulator initially modeled and implemented in MATLAB [8], which has been substantially modified in this work in order to met our project requirements. So a brief about this simulator is given in the next. It implements a 4-DOF sailboat model, which differs from other models by taking into account changes in the roll angle. It has three main blocks: control, model and interface. The control block implements a simple heading controller, which makes the boat follow a desired heading. The heading controller receives a reference orientation as input, finds the error between the current sailboat heading and the reference and then uses this error to adjusts the rudder. The modeling block receives an initial state of the system and computes the following states, according to the modelled equations for the sailboat movement. Finally, the visualization block provides a graphical interface, allowing the user to observe the sailing behavior throughout the simulation. To the purpose of using the above simulator for obtaining our intents, we performed several changes on it. One of the changes is the replacement of the original control block for the PI heading controller initially developed for the N-Boat I [10].

III. Short-term Path Planning with Upwind

In order to devise a general purpose algorithm, the three maneuverings described in Section II-A should be taken into account. In this work, the wind direction can be determined in relation to the sailboat and the target direction using a windsock sensor. Based on this, the best strategy can then be chosen from the above ones in order to get heading to the desired target. That is simple if the target is not in the closed hauled region, a straight control can be performed on the rudder and sail that can bring it to the goal. However, another path should be devised (and optimized) when the target is not straight reachable. In this case, some quantity of points has to be calculated, alternated on one and another side of the target direction, in order for the boat to do a zigzag passing over them. How much points to use and the tacking angles depends on the task, if the boat has to reach the goal faster or slower.

So, basically, this upwind path planning method must find points in-between the starting and target points, allowing the sailboat to reach a desired point that is directly against the wind. The previous heading control strategy implemented in the N-Boat I would work properly except when the target is in the dead area. The solution proposed here for this problem is to find a mathematical model that generate alternating points that are reachable using the described control strategy. Figure 5 illustrates the proposed mathematical model. Initially, our method finds the line A using the actual $P_0 = (x_0, y_0)$ and target $P_d = (x_d, y_d)$ points, with solution given by Equation 1.

$$a_A = \frac{y_d - y_0}{x_d - x_0}; \qquad b_A = y_0 - a_A x_0 \tag{1}$$

Next, a line B is found that has a θ_t angle with the line A, using Equation 2. Notice that the angle θ_t controls the

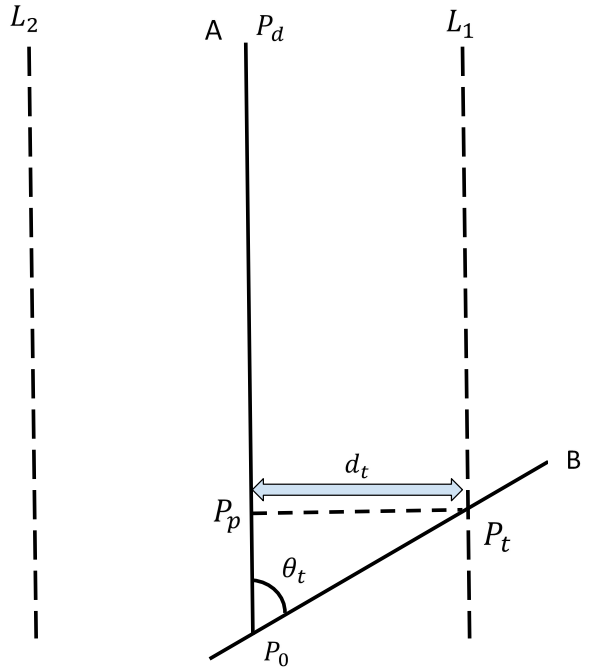

Fig. 5. Diagram showing the model used for the path planing method. P_0 and P_d are the start and target points, respectively. d_t and θ_t are the tacking distance and orientation.

sailboat heading during the maneuver so it must be carefully chosen to keep the sailboat away from the dead area.

$$a_B = \frac{a_A + \tan(\theta_t)}{a_A \tan(\theta_t) - 1}; \qquad b_B = y_0 - a_B x_0 \tag{2}$$

The next step is to find a point P_t in line B so that the Euclidean distance between P_t and P_p is d_t. This point P_t is used in Equation 3 to find the lines L_1 and L_2 that are parallel to line A (defining a limiting region).

$$b_{L_1} = y_t - a_A x_t; \qquad b_{L_2} = 2b_A - b_{L_1} \tag{3}$$

Equation 4 is used to find the quantity k of steps with size $d_{0,p}$ to go from P_0 to P_d, also defining the number of tacking points.

$$k = \frac{d_{0,d}}{d_{0,p}} \tag{4}$$

Equation 5 gives the X and Y steps to advance a distance $d_{0,p}$ in the P_0-P_d direction.

$$\Delta x_{p,0} = x_p - x_0; \qquad \Delta y_{p,0} = y_p - y_0 \tag{5}$$

At this point, it is possible to find the (x, y) coordinates of all tacking points. Starting at P_0, advances with $n\Delta x$ and $n\Delta y$ step, where $n = 1, 2, \ldots, k$. The tacking points are the projections, at each step, of the point in the line L_1 in even steps switching to line L_2 in odd steps. Note that parameters θ_t and d_t are responsible for changes in the method behavior. Different values of them imply in different tacking points.

Algorithm 1: SHOT-TERM PATH PLANNING METHOD

Input: $P_0, P_d, \theta_t, d_t, \theta_{v_B}, \theta_{d_W}, \theta_{v_W}$
Result: Matrix N containing the trajectory points

begin
  if $\theta_{v_B} < 40$ **then**
    $a_A \leftarrow \frac{y_d - y_0}{x_d - x_0}$; $b_A \leftarrow y_0 - a_A x_0$;
    if $\theta_{v_W} - \theta_{d_W} < 0$ **then**
      $a_B \leftarrow \frac{-a_A + \tan(\theta_t)}{-a_A \tan(\theta_t) - 1}$;
    else
      $a_B \leftarrow \frac{-a_A - \tan(\theta_t)}{a_A \tan(\theta_t) - 1}$;
    end
    $b_B \leftarrow y_0 - a_B x_0$;
    $x_p \leftarrow x_d$; $y_p \leftarrow y_d$;
    repeat
      $x_p \leftarrow \frac{x_p + x_0}{2}$; $y_p \leftarrow \frac{y_p + y_0}{2}$;
      $P_t \leftarrow Proj_{p,B}$;
      $d \leftarrow d_{p,t}$;
    until $d \leq d_t$;
    $b_{L_1} \leftarrow y_t - a_A x_t$; $b_{L_2} \leftarrow 2b_A - b_{L_1}$;
    $\Delta x \leftarrow x_p - x_0$; $\Delta y \leftarrow y_p - y_0$;
    $P_{0_{L_1}} \leftarrow Proj_{m,L_1}$;
    $P_{0_{L_2}} \leftarrow Proj_{m,L_2}$;
    $k \leftarrow \frac{d_{0,d}}{d_{0,p}}$;
    for $z \leftarrow 1$ *to* $k - 1$ **do**
      $\Delta x_t \leftarrow z\Delta x$; $\Delta y_t \leftarrow z\Delta y$;
      if $mod(z, 2) = 0$ **then**
        $N_{z,1} \leftarrow x_{0_{L_1}} + \Delta x_t$;
        $\mathrm{N}_{z,2} \leftarrow y_{0_{L_1}} + \Delta y_t$;
      else
        $N_{z,1} \leftarrow x_{0_{L_2}} + \Delta x_t$;
        $\mathrm{N}_{z,2} \leftarrow y_{0_{L_2}} + \Delta y_t$;
      end
    end
  end
end
Output: N

IV. TACKING METHOD OPTIMIZATION

For the tacking method presented in section II-A to work properly, parameters θ_t and d_t should be carefully chosen. Variation in these parameters can lead to faster trajectories, wider maneuver area, or even points can be created that make the sailboat not to reach the target as shown in the top graphic of Figure 6. During a beating maneuver, the parameter that most influences the arrival time is the number of tacks [2]. During tacking, the sailboat goes through the dead area thus loosing some amount of its velocity. Ideally, it looks simple choosing optimal tacking points so that arrival time is reduced. In fact, it is empirically proven that the best strategy is to pick up the smallest amount of tacking points, which is just one point, but that is not always feasible nor possible.

Initially, we have done a series of experiments to identify the kind of function to optimize. The idea here is to use

Algorithm 2: PATH PLANNING OPTMIZATION ALGORITHM

Input: genetic algorithm parameters
Result: The optimal θ_t and d_t

```
begin
    genetic algorithm parameters initialization;
    generate initial populations randomly;
    for i ← 1 to N_ger do
        foreach individual do
            use the individual's θ_t and d_t in the
              simulation and find the arrival time;
        end
        compute the fitness of each individual;
        aply roulette selection to population;
        aply crossover to selected individuals;
        aply mutation to the new individuals;
        produce new population;
        store the best individual;
    end
end
```

Output: θ_t, d_t of the best individual

a brute force method to find the optimal values of θ_t and d_t. This brute force method consists in simulating each peer (θ_t, d_t) varying the parameters θ_t from 20 to 80 and d_t from 10% to 90% of the initial distance to target. Figure 8 shows the arrival time for one of these executions. One can see the existence of several local minima. Hence, genetic algorithm can be applied here as a good alternative to find the global minima approximation. The issue is that the problem in focus has a very large search space. Each variation in speed or wind direction, heading, and initial sailboat speed results in different arrival times in the simulator. In practice, the boat would apply this method at the beginning of each maneuver when a waypoint is set. The system checks its status and uses it as the initial condition in the simulator and thus apply the proposed methodology. So our approach breaks the problem in some cases, analyzing the performance and validity of the method for these cases. We remark that once the method is verified and validated the system as a whole can be embedded in the N-Boat II for further practical experiments.

A conventional genetic algorithm is used to optimize the path planning method as described in Algorithm 15. Individuals are represented by binary strings containing information about θ_t and d_t. An initial population with n individuals is randomly generated. The calculation of the fitness of individuals in the population is performed in the simulator. Each individual in the population has their values θ_t and d_t simulated. The fitness is then represented through the spent time, so that individuals with lower arrival times gets higher fitness. Part of the population is then selected for the crossover step using the roulette, where individuals with higher fitness have a higher probability of being selected. To make a cross cut, points are selected for each portion representing the binary θ_t and d_t. And then the exchange is performed in the tails. Mutation probability is tested for every bit of each individual. Each bit has a certain probability of suffering mutation, which consists in changing the bit value. A new population is formed, containing individuals generated by reproduction and part of the best individuals from the last population. This process is repeated until a desired number of generations is met as is illustrated in Algorithm 15.

V. EXPERIMENTAL RESULTS

A series of experiments are performed in order to verify and validate the behavior obtained by changing the parameters of the tacking method.

A. Experiments with the path planing method

Without loss of generality, the boat starts in $(0, 0)$ with $\theta_b = 0$ degrees and initial velocity of 2m/s. The target is set to $(0, 200)$ that is directly against the wind. In the experiments shown in Figure 6, the d_t parameter is set at 25m and θ_t varies, representing a situation where the sailboat movement is limited. It is possible to note in this case that the higher the value of θ_t, the greater the number of points generated by tacking method. In the experiments of Figure 9 the value of θ_t is set at 50 degrees and tacking distance is varying, without maneuvering restrictions. With increasing distance, the less tacking points are needed to generate the path in order to keep the sailboat to 50 degrees (against the wind).

It is important to note in the results the diversity of tacking points that can be generated by the method to reach the same target. The experiments also shows that the trajectories with fewer tacking points are those with the lowest arrival time.

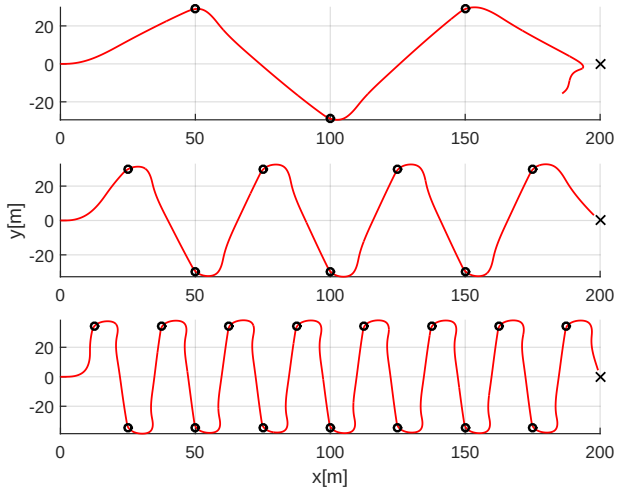

Fig. 6. Upwind trajectory obtained in simulation with constant d_t and varying θ_t (red line is the sailboat trajectory). Black crosses are the target points and black circles are the points generated by the path planing method. The parameters $[\theta_t, d_t, t_a]$ are $[30, 25, 244.8]$ for the upper graphic, $[50, 25, 172.8]$ for the middle graphic and $[70, 25, 134.4]$ for the bottom graphic.

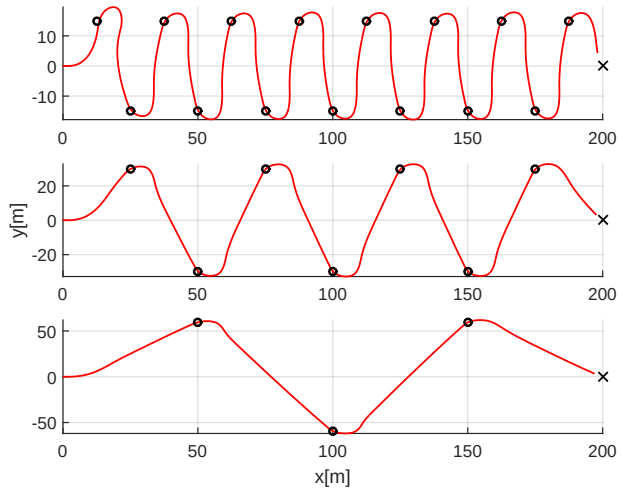

Fig. 7. Upwind trajectory obtained in simulation with varying d_t and constant θ_t (red line is the trajectory). Black crosses are the target points and black circles are the points generated by the path planing method. The parameters $[\theta_t, d_t, t_a]$ are $[50, 15, 191.2]$ for the upper graphic, $[50, 25, 172.8]$ for the middle graphic e $[50, 50, 368.8]$ for the bottom graphic.

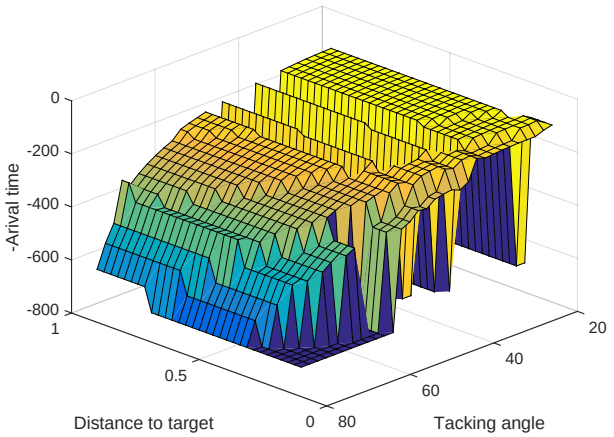

Fig. 8. Brute force method graph obtained in simulation with varying parameters d_t and θ_t.

TABLE II

SIMULATION PARAMETERS FOR EACH CASE

Cases	Wind direction - degrees	Wind velocity - m/s
1	60	10
2	90	15
3	60	15

B. Optimizing the path

In the following, experiments are performed in order to compare the performance of tacking optimization method against the brute force method above. To validate this method it is necessary to separate the problem in cases. For each case, the optimization method and the brute force are executed to find its best arrival times. In the experiments with the brute force method the parameters θ_t and d_t are discrete. After the simulations, it is possible to find what parameters generate the lowest arrival time for each case. A graphic is generated to show the influence that changing the parameters has on the arrival time. In another set of experiments to test the optimization method, it is applied ten times to each case. At the end of all computations, the average execution and arrival times are calculated. Table II shows the wind conditions used in each case.

By analyzing the results of Table III we can see that the method worked satisfactorily finding points similar to those obtained in the brute force method and drastically reducing the execution time. In cases 1 and 3 the optimization method could even find better points than the brute force method. This is possible because the optimization method hit search area points that have not been achieved by the method of brute force thanks to the precision used. If the precision of the brute force method is increased, the simulation time would be too high. Case 2 is the only one where the optimization method has not obtained a better result than the brute force method. Anyway, the result is 91.22% of the best value obtained by brute force.

VI. CONCLUSION

This work proposes a new analytical approach to the problem of automatic short term path planning in upwind sailing and its optimization. The results verify the proper working of the short term path planning method, allowing the sailboat to reach the target overall cases. The existence of two parameters (θ_t and d_t) gives versatility to method. This can allow a higher level agent to chose the sailboat behavior during a beating maneuver such as keeping the sailboat in a safe area or changing the tacking angle to increase sailing stability. The optimization method is able to produce similar or better results when compared to the brute force approach, being about ten times faster.

Future work (already started) consists on the implementation of these methods in a larger sailboat to allow further experimental testing and validation in a real situation. After this it will be possible to reduce the computational complexity of time of the optimization method, allowing its real-time, on-line usage in the N-Boat II - The Robotic Sailboat.

ACKNOWLEDGMENT

REFERENCES

[1] C. Pltrs, M. Romero-Ramirez, F. Plumet and B. Alenssadrini, Modeling and reactive navigation of an autonomous sailboat, presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, Semptember 25-30, 2011.

TABLE III

OBTAINED RESULTS

Cases	Method	Best arrival time	Simulation time
1	Brute force	102.4	2364.3
	Genetic algorithm	97.72	176.48
2	Brute force	105.2	2172.42
	Genetic algorithm	114.44	216.28
3	Brute force	82	2559.96
	Genetic algorithm	80.16	196.32

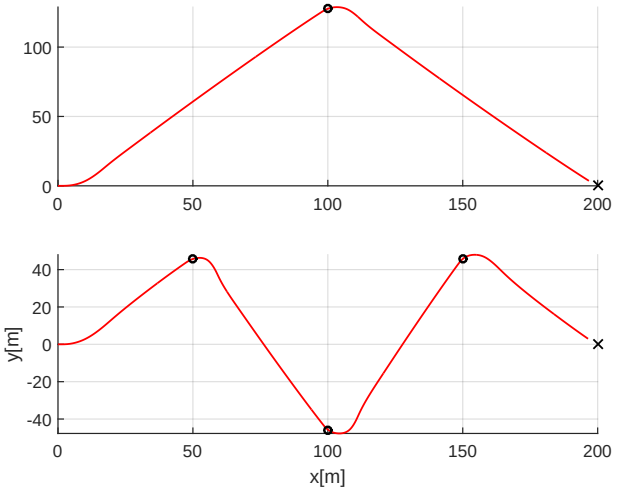

Fig. 9. Comparison between the best tacking points found with the brute force method and the optimization for case 2. The parameters $[\theta_t, d_t, t_a]$ are $[51.96, 180, 105.2]$ for the upper graphic and $[42.05, 80, 114.44]$ for the bottom.

[2] R. Stelzer, Autonomous Sailboat Navigation: Novel Algorithms and Experimental Demonstration, Ph.D. thesis, Centre for Computational Intelligence, De Montfort Univ., Leicester, UK, 2012.

[3] Williamsz, Ben et al. Senior Project Design of a Two Meter Autonomous Sailboat. Proceedings of the 2014 International Technical Meeting of The Institute of Navigation, San Diego, CA, USA, January 27-29, 2014.

[4] N. A. Cruz and J. C. Alves, Navigation Performance of an Autonomous Sailing Robot, Proceedings of the MTS-IEEE Conference Oceans'2014, St. John's, Canada, September 2014.

[5] A. Philpott and A. Mason, Optimising Yacht Routes under Uncertainty, presented at the 15th Chesapeake Sailing Yatch Symposium, San Francisco, Annapolis, USA, July 22-26, 2001.

[6] D. S. Ferguson and P. Elinas, A Markov Decision Process Model for Strategic Decision Making in Sailboat Racing, presented at Canadian AI 2011, Toronto, Canada, Febuary 13-18, 2011.

[7] R. C. Dalang, et al. Stochastic Optimization of Sailing Trajectories in an Upwind Regatta, published in Journal of the Operational Research Society, Vol. 66, Issue 5, pp. 807-821, May, 2015.

[8] L. Xiao and J. Jouffroy, Modeling and Nonlinear Heading Control of Sailing Yachts, published in IEEE Journal of Oceanic Engineering,, Vol. 39, NO. 2, April, 2014.

[9] D. H. dos Santos, Project and Implementation of a Low-level Controller for Robotic Sailboats (In Portuguese). Monography for Final Project of Undergraduate Course - Computer Engineering at Federal University of Rio Grande do Norte, Natal, RN, Brazil, December, 2014.

[10] D. Santos, A. Silva Junior, A. Negreiros, J. Vilas Boas, J. Alvarez, A. Araujo, R. Aroca, and L. Gonçalves. Design and Implementation of a Control System for a Sailboat Robot, published in Robotics 2016, Vol. 5, NO. 1, February, 2016.

[11] David Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley Professional. ISBN 978-0201157673, 1989.