A Reactive Approach to Obstacle Avoidance in Autonomous Sailing

Roland Stelzer*†, Karim Jafarmadar*, Hannes Hassler*, and Raphael Charwot*
*INNOC - Austrian Society for Innovative Computer Sciences
Kampstraße 15/1, 1200 Vienna, Austria
http://www.innoc.at, http://www.roboat.at
Email: {roland.stelzer,karim.jafarmadar,hannes.hassler,raphael.charwot}@innoc.at
†Center for Computational Intelligence
School of Computing at De Montfort University
The Gateway, Leicester LE1 9BH, United Kingdom
http://www.cci.dmu.ac.uk

***Abstract*—This paper presents a reactive approach to obstacle avoidance for autonomous sailboats. It is an extension to the short course routing method published by Stelzer and Pröll in 2008 [1] which enables it to deal with obstacles in real-time. First simulation results are promising. The algorithm enables an autonomous sailboat to circumnavigate differently sized obstacles under various wind conditions successfully.**

I. Introduction

Robotic sailing boats execute the complex sailing processes completely autonomously and without human interaction. Starting with the calculation of the optimal route based on weather data, to the autonomous execution of manoeuvres like tack and jibe, robotic boats are able to reach any desired destination by analyzing sensor data through artificial intelligence.

An important problem to be solved for long-term unmanned and autonomous missions on sea is reliable obstacle detection and avoidance. Static obstacles such as landmasses can be predefined on the sea map as a basis for the routing system. A combination of multiple techniques, such as thermal imaging, radar, camera, and automatic identification system (AIS) can be used to detect dynamic obstacles. Research in this field has been carried out for autonomous underwater vehicles [2] and motorised autonomous surface vehicles [3]–[6]. The obstacle avoidance task is different for sailing vessels, as they can not navigate in any direction directly, depending on wind conditions. Therefore a more sophisticated approach to autonomous obstacle avoidance is presented.

A. ASV Roboat

The approach described here has been implemented in the control system of the *ASV Roboat*[1] (Fig. 1) which has been in development by a research team of the Austrian Society for Innovative Computer Sciences (INNOC) since 2006.

The basis for the *ASV Roboat* is the commercially available boat tyle Laerling[2], a sailboat designed by Jan Herman Linge. The boat was originally created for kids to learn sailing, and therefore safety and stability are its major characteristics. It has

[1]http://www.roboat.at
[2]http://www.laerling.nl

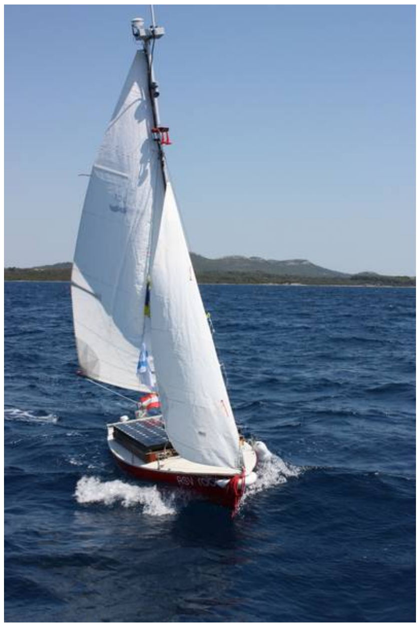


Fig. 1. The ASV Roboat autonomous sailing vessel (ASV)

a length of 3.75 m and comprises a 60 kg keel-ballast, which will bring the boat upright even from the most severe heeling. The boat can carry large payloads such as a battery bank and multiple sensors. Including batteries the overall weight of the boat is about 300 kg. Additional payload of up to 50 kg is possible without significant impact on the sailing behaviour. The sail area of mainsail and foresail together is 4.5 m^2. It is equipped with solar panels providing up to 285 W of power during conditions of full sun and a direct methanol fuel cell delivering 65 W as a backup energy source. The *ASV Roboat* features a three-stage communication system, combining WLAN, UMTS/GPRS and an IRIDIUM satellite communication system, allowing continuous real-time access from shore [7]. This can be used, for example, to monitor the ship, or to transmit tartget coordinates to the boat. The rudder and sails as well as the tacks and jibes are autonomously controlled by a Linux-based onboard computer system using incoming data from various sensors (GPS, compass, anemometer, etc.) on an NMEA2000-bus.

The Austrian team of scientists has proven itself in numerous competitions. The Roboat-team won the first international Microtransat event [8] in Toulouse, France in June 2006. In September 2007 the team won the Microtransat again on the Irish Sea in Aberystwyth, Wales. And in May 2008 the *ASV Roboat* became first World Champion in robotic sailing on Lake Neusiedl, Austria. Most recently, the team defended their title at the World Robotic Sailing Championship[3] which took place off the Atlantic coast in Portugal. Efforts are being made to utilise the autonomous sailing technology for concrete research purposes. Preparations are currently underway for a project to research the whale population in the Pacific Ocean [9].

B. Integration in a Layered Architecture

Mobile autonomous robot systems are usually divided into separate layers each responsible for a part of the problem. Basically two different architectures exist: topdown planner based and bottom-up reactive systems. In Top-down planner-based systems sensor data together with a priori knowledge about the environment is used to generate a model of the world in which planning occurs [10], [11]. Planning mechanisms generate a detailed plan for the robot's actions to reach a previously specified goal. After the plan is finished the robot will act according to it. These systems have shown good performance in complex static environments, however generating a plan is usually a time intensive task. Thus these systems cannot react quickly in dynamically changing and unpredictable environments. The bottom-up reactive approach connects measured sensor data directly with the robot's actuators [12]. Therefore the robot can respond fast to changes in the world like unexpected and moving obstacles. Reactive or behaviour-based robots have shown great performance in constantly changing environments often found in real world tasks. Since the robot only acts on local information without global knowledge about the environment it may not reach a global optimum and often lacks the ability to perform complex tasks. These two approaches can be combined in an hybrid architecture [13]–[16].

Such a hybrid multi-layer control architecture is used for the *ASV Roboat* combining both reactive and planner-based approaches. The control system is divided into four layers (Fig. 2). Each layer has access to sensor data by connecting to the data abstractor. The *Abstractor* is a computer program which is executed directly on the boat. It gathers sensor data and transforms the raw data into semantically useable values. Preprocessing like damping, scaling, unit transformations, or plausibility checks are done at this level. A detailed description of each layer and the interaction between adjacent layers can be found in [17].

The strategic long term routing layer determines an optimal rough route with respect to the boat-specific behaviour, the

[3]http://www.roboticsailing.org

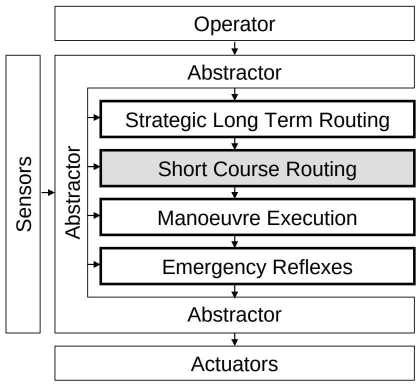


Fig. 2. ASV Roboat system architecture: reactive obstacle avoidance is part of the short course routing layer

predicted weather conditions and sea topology. The route is automatically divided into many short legs and described as an ordered set of coordinates to be passed. The next target coordinate is handed on to the layer below, the *Short Course Routing* layer.

The presented approach to obstacle avoidance is an extension to the current *Short Course Routing* layer implementation in the *ASV Roboat*. A short summary of this routing algorithm published by Stelzer and Pröll in 2008 [1] is given below.

II. Obstacle Avoidance Algorithm

A. Short Course Routing

1) Sailboat Behaviour (Polar Diagram): The actual speed a sail boat can reach in a certain direction depends on the wind speed but also on the angle between boat heading and wind direction: while no direct course is possible straight into the wind, the maximum speed is usually obtained with the wind from the rear side at about $\pm 120deg$. This dependency can be plotted continuously as the boat-specific polar diagram (Fig. 3).

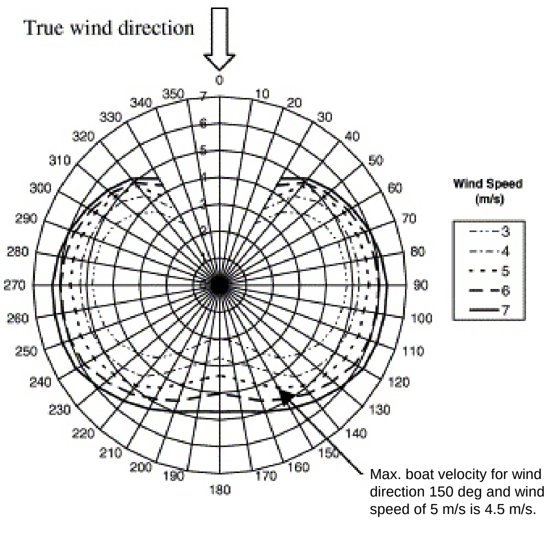


Fig. 3. Example of a polar diagram [18]

The boat speed $\vec{v}_b$ is therefore given as a function of the true wind speed $\vec{w}_{abs}$ and the angle between true wind and boat heading:

$$|\vec{v}_b| = f\left(|\vec{w}_{abs}|, |\varphi(\vec{v}_b) - \varphi(\vec{w}_{abs})|\right) \tag{1}$$

2) Quantification of Target-Approach (Velocity Made Good): In order for the boat to get from a current position B to a target point T, both the direction of the target and the wind must be considered. The aim of the routing algorithm is therefore to decrease the distance to the target as fast as possible. The efficiency of a certain boat heading in approaching the target v_t can be directly quantified projecting the boat speed vector $\vec{v}_b$ on the target direction $\vec{t_0}$ as illustrated in Fig. 4(a):

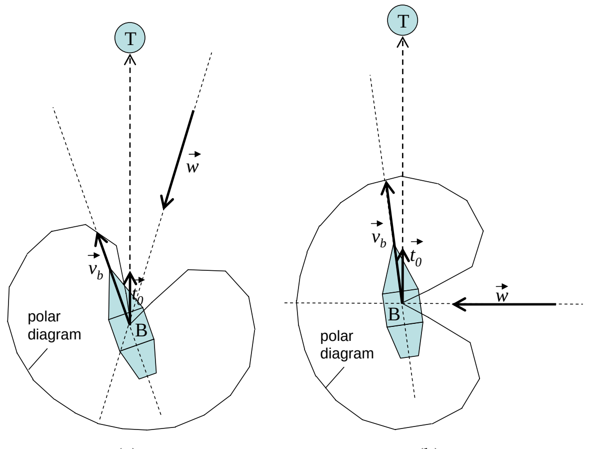
$$v_t = \vec{v}_b \cdot \vec{t_0} \tag{2}$$

The boat speed vector $\vec{v}_b$ can be considered to be a function of the boat heading $\vec{v}_{b,0}$ and the wind vector according to Eq. (1). The unit vector $\vec{t_0}$ indicates the direction from the current boat position B towards the target T. If the target is located in the direction the wind comes from, the optimal route is a compromise between aiming towards the target and getting speed. The goal for the routing algorithm is to identify the boat heading for which the velocity made good v_t, which represents the negative time-derivative of the distance between boat and target, is maximised. The same approach works if the target is located in any direction relative to the wind direction (Figs. 4(b), 4(c)). However, the optimal boat heading indicated by the direction of the speed vector changes as the boat moves on its trajectory. The situation in Fig. 4(b) promises unique optimum boat heading until the target is reached and the steady correction of the boat heading is smooth along the trajectory. The situations in Figs. 4(a) and 4(c), however, will lead to constellations where there are two headings of equal maximum velocity made good to follow, one on the right and one on the left hand side of the wind direction. This happens when the target direction aligns with the wind direction (Fig. 5). In order to get a unique proposal for the heading to follow, a hysteresis condition is applied.

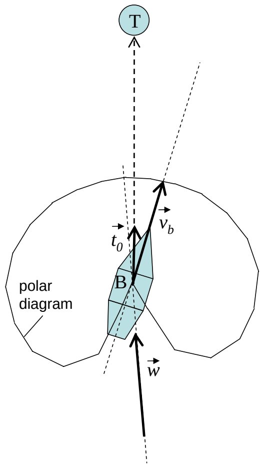
3) Beating Hysteresis and Beating Parameter: In practice, the sailor beats about if the target is within the angle where no direct navigation is possible. In the terms of our analytical approach, this means that the boat follows a local optimum $\vec{v}_b$ (close to the recent heading) for a certain time until the global optimum $\vec{v}_b'$ is significantly better than $\vec{v}_b$. At this point, the boat turns for the global optimum $\vec{v}_b'$, which will be followed until an alternative heading is significantly better leading to the next turn and so on. A hysteresis factor n is defined by:

$$v_t' > n \cdot v_t \rightarrow \text{turn for } \vec{v}_b'; n > 1 \tag{3}$$

In order to obtain a reasonable behaviour of the algorithm, n must be larger than one.



(a) (b)



(c)

Fig. 4. Possible constellations and definition of velocity made good

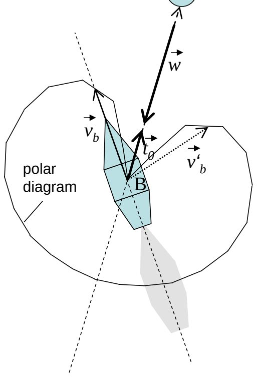


Fig. 5. Two global optima for target efficiency

B. Routing Extension for Obstacle Avoidance

To deal with obstacles we extend our *quantified target approach* (depicted in Fig. 4 and 5) by an *obstacle quality*. There is an overall maximum distance which we call the *safe horizon* r_{max}. Segments beyond it can be ignored. Obstacles within the safe horizon put a penalty on the directions leading to them. This penalty increases the closer an obstacle is. Very close obstacles whose distance threaten to fall below a certain value r_{min} mark corresponding directions unnavigable.

We distinguish two cases:

- **Unnavigable** courses which cannot be navigated because of wind direction or because an obstacle is too near (see Section III-D). These courses are not considered in the calculation of an optimal course.
- **Navigable** courses. Considering near obstacles (distance $d_b < r_{max}$ safe horizon) the courses are dynamically modified. Directions leading to an obstacle suffer a penalty q_b as a weight to v_t discussed in Eq. 2. q_b is calculated by formula

$$q_b = \min\left(1, \max\left(0, \frac{d_b - r_{min}}{r_{max} - r_{min}}\right)\right) \qquad (4)$$

in effect a linear scaling between 0 and 1 within the range $r_{min} \leq d_b \leq r_{max}$ and 0 or 1 outside of it respectively; see Fig. 6

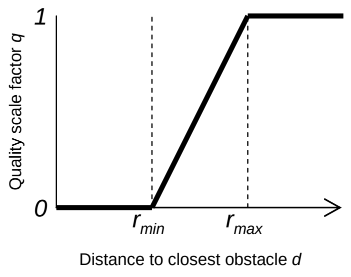


Fig. 6. Linear scaling on polar diagram according to distance to obstacles

Distinguishing these two cases and using the qualifying weight from the latter we establish a *qualified* v_t^* (in comparison and extension to v_t from Eq. 2):

$$v_t^* = \begin{cases} -\infty, & \text{if } |\vec{v_b}| = 0 \\ -\infty, & \text{if } r_{min}\text{-violation (see Section III-D)} \\ q_b \vec{v_b} \cdot \vec{t_0}, & \text{else} \end{cases} \qquad (5)$$

A high penalty means a low q_v (close to 0), a low penalty means a value close to 1. This qualified v_t^* allows the course optimization algorithm to compare values of different courses where obstacles are included in the metric as well.

In Fig. 7 we show the dynamic modification of the polar diagram. Only obstacles within the safe horizon r_{max} are considered, which omits obstacle O_4. Obstacle O_1 puts only a small penalty to the direction vectors leading to it. The closer obstacle O_2 causes a much higher penalty on its directions.

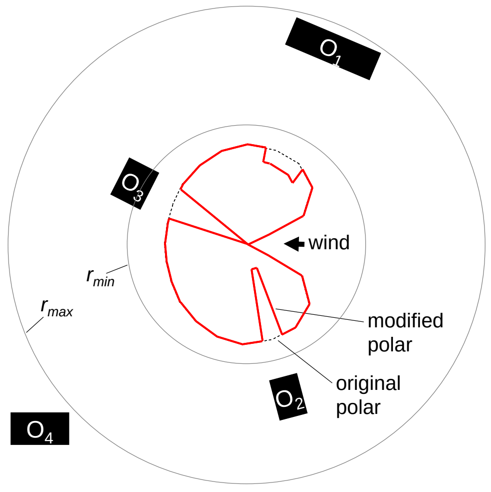


Fig. 7. Influence of obstacles O_1-O_4on polar diagram

This results in a dent in the polar diagram. Obstacle O_3 violates r_{min} and therfore the directions towards it get marked unnavigable.

III. Implementation

A. Obstacle Data Processing

In collision avoidance a robotic sailboat must deal with its nearest obstacles. We abstract the boat to a point of view and the obstacles to polygons consisting of line segments. Rays radiating from the point of view, for instance as discrete angles from 0 deg to 359 deg, divide the surrounding area into sectors of equal size. For each sector we determine the minimum distance to the nearest obstacle.

In our context, the result of this calculation is used for close range course planning. For 360 rays the result would be returned as an array of 360 numbers, each one representing the corresponding distance to its nearest obstacle. We call this data structure the *all-around-array*, *AAA* for short. At elements without a value, respectively the maximum value initially set, there are no obstacles within the safe horizon. In Fig. 8, which illustrates a simplified AAA of 16 elements, we depict the the minimum distance to the nearest obstacle.

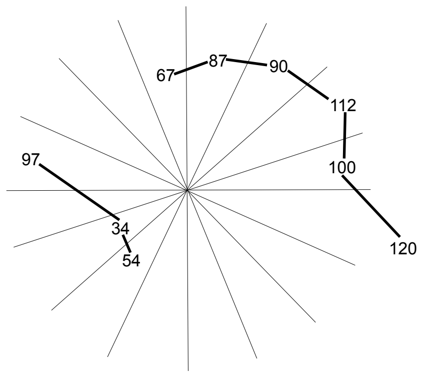


Fig. 8. AAA-Values. Sectors without obstacles have no value respectively safe horizon r_{max} as default)

B. Weeding Out Non-Relevant Data (Culling)

Only obstacles that are closer than r_{max} are considered relevant for collision avoidance. Because total number of obstacles can be very large it is necessary to implement object culling in an efficient way. To avoid frequent distance calculations between the boat and every known object, the algorithm uses the following caching mechanism:

At the beginning all obstacles o_j are sorted based on their distance $d(o_j, S)$ to the starting point S. Trivially, all initially relevant objects lie within $0 < d(o_j, S) < r_{max}$. After the boat has traveled a distance L the possibly interesting obstacles can be retrieved by querying the previous list for all objects for which

$$L - r_{max} < d(o_j, S) < L + r_{max} \tag{6}$$

This formula describes a doughnut with radius L and gauge $2r_{max}$ as shown in Fig. 9.

The resulting range of objects can be further reduced by taking the moving direction into consideration as indicated in the illustration. Because the area of the doughnut grows with L, the sorting should be repeated relative to the current position S' whenever L exceeds a certain threshold.

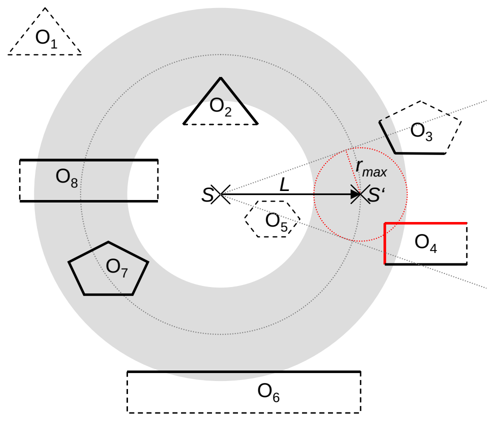


Fig. 9. After traveling a distance of L, all now relevant objects lie within the gray area.

C. Sort and Sweep Algorithm

1) Description: For each sector only the distance to the closest obstacle is relvant. To calculate the minimum distance per sector we have to evaluate the obstacle polygons considered to be in relevant distance to the boat. Sort-And-Sweep assumes that no two line segments intersect with each other. This is the case when the line segments are used as building blocks for non-overlapping simple polygons, not intersecting on itself.

Sweep line algorithms are a major technique in algorithmic geometry. They led 1976 to a breakthrough in the computational complexity of geometric algorithms when Shamos and Hoey presented algorithms for line segment intersection in the plane [19]. The technique itself may be traced to scanline algorithms of rendering in computer graphics. Wylie et al. introduced the scanline rendering technique in 1967 [20].

(a) Sort all points of the line segments radially, i.e. ascending according to their angular position relating to the boat position. We call this the *points angular list*. Each point in this list keeps a link to its line segment.

(b) Take the first point of the *points angular list* and put the according line segment into the *list of current scan (LOCS)*, which is sorted by distance from the center.

(c) Get the next point from the *points angular list*. This point can be either a beginning point of a new segment or an ending point of a sement. In the fist case put the according segment into the LOCS. LOCS is maintained sorted. In the other case the according segment is removed from LOCS. Possibly a new line segment becomes the element with minimum distance to the center. Step (c) is repeated for each point of *points angular list*.

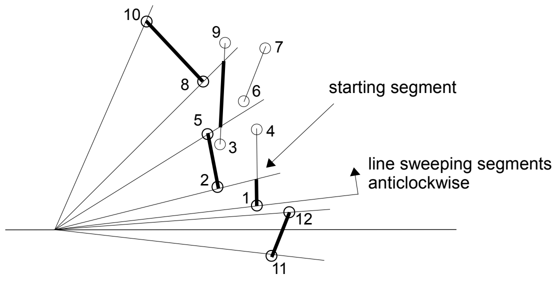


Fig. 10. Sort and sweep example

Fig. 10 illustrates the *sort and sweep* algorithm with the help of an example. The sweeping starts at the point with the smallest angle. Only segment completely above the abscissa are considered for the first starting point. Element changes of LOCS are indicated by small circles around the respective points. Not all points bring about a new minimum LOCS-element. At points 3, 4, 6, 7, and 9 the LOCS is changed but its minimal segment part stays in front. A useful technique for dealing with segments passing the abscissa (here line segment 11 to 12) is to overshoot the small positive angle by adding 360 degrees to it (point 12). This guarantees a satisfying sorting order in Step 1 and such a segment will be dealt with later.

2) Analysis: The non-intersection property guarantees that the number of points is the maximum number of executions of steps (c). LOCS maintains its sorting order during the sweep from one point to the next. Although the distance of each segment, especially of the closest segment, will change.

Step (a) needs $\mathcal{O}(n \log n)$ time (n the number of line segments), Step (b) $\mathcal{O}(1)$ and Step (c) needs for per manipulation of LOCS (insertion or deletion) $\mathcal{O}(\log n)$ and is repeated $2n - 1$ times. Therefore the whole algorithm is bounded by $\mathcal{O}(n \log n)$ as well.

D. Minimal Distance Maintenance (Flower Algorithm)

In order to ensure a minimal safety distance r_{min} to any obstacle, additional evaluations are carried out. Just the modification of the polar diagram according to the AAA (see Fig. 7) is not sufficient. For instance, the boat would follow a route close to parallel along a straight obstacle line.

Looking into the direction of the route the distance might be larger than the allowed r_{min}, however the side-distance can still become less than that by following this route. Therefore, it is neccessary to evaluate additionally whether a certain sector of *AAA* is allowed with respect to the other sectors. Assuming there is no obstacle within a radius r_{min} we can choose a test point in each sector with a distance of r_{min} from the current boat position. Only segments with a distance less or equal than $2r_{min}$ to the boat are critical and need to be considered. A r_{min}*-violation* is detected, if one critical segment would reach a distance less than r_{min} to one of the sector test points. In this case,the corresponding sector is marked forbidden and must not be chosen for the next route decision.

We call this the *flower algorithm* because the “blossoming” of r_{min} circles reminds of petals. its complexity is $\mathcal{O}(\#\textit{critical segments} \cdot \#\textit{AAA-sectors})$.

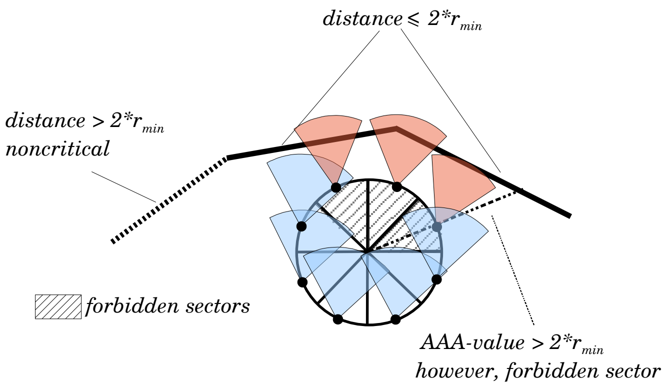


Fig. 11. Example for mimimal distance mainainance (flower algorithm)

Fig. 11 illustrates and example of the flower algorithm. The boat is considered to be in the center of the 8-sectored circle with radius r_{min}. Only segments which are closer than $2r_{min}$ are critical and have to be evaluated for each sector-test-point (small knots on the r_{min}-circle around the boat). Note the north-east-east-sector: it has an AAA-value of more than $2r_{min}$ (dashed line), however its test-point gets into a less than r_{min} neighborhood. Therefore, this sector is marked forbidden. All sectors with its test-points closer than r_{min} to any obstacle line (red beacons) are marked forbidden and must not be entered in the boats next move. Sectors with a blue beacon (i.e. not cutting into any segment) are safe. For illustrative purposes, the beacons are drawn here only into the direction of the critical sectors.

IV. Experiments and Results

A. Experimental Setup

Experiments have been carried out using the same software implementation of the navigation algorithm which is used on *ASV Roboat*, but in a simulated environment. For the simulation a simple boat model is used, where rudder and sail movements are neglected and the simulated boat always follows the calculated heading with optimal speed. Fig. 12 shows the simplified polar diagram which is used in the simulation to describe the boat behaviour. All directions are equally rated, except a 120 deg no go zone upwind which is set to 0.

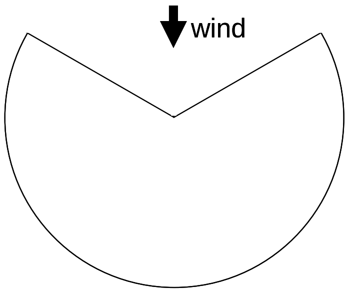


Fig. 12. Simplified polar diagram

For all experiments the overall distance between start and target was 1000 m. An obstacle, represented by a single line, was placed midway. The size of the obstacle was 50 m respectively 200 m. The values of the parameters are shown below:

- $r_{min} = 50$ m
- $r_{max} = 250$ m

B. Scenarios

Three scenarios with different courses where simulated. The simulated trajectories can be found in Fig. 13 for beam reach course, Fig. 14 for upwind course and Fig. 15 for downwind course respectively. All three scenarios where carried out with small (Fig. 13(a), 14(a), 15(a)) and large obstacles (Fig. 13(b), 14(b)-14(c), 15(b)-15(c)).

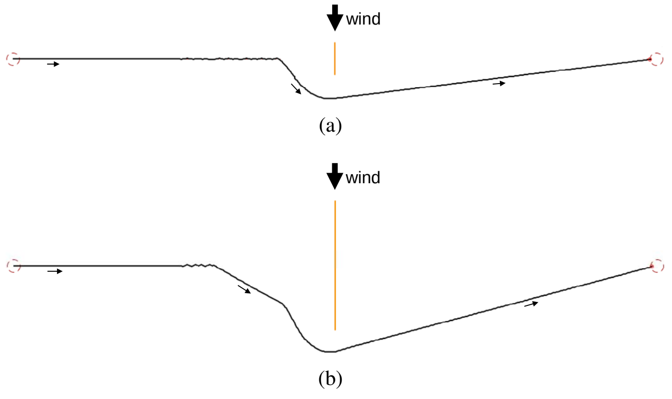
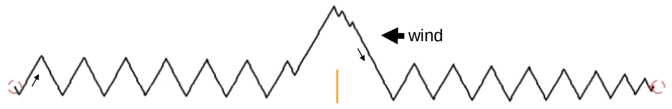


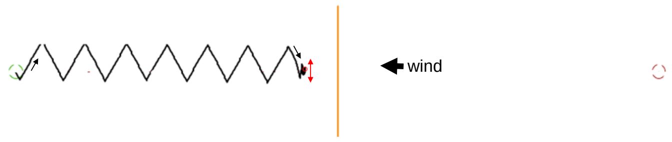
Fig. 13. Simulation results in a beam reach

C. Results

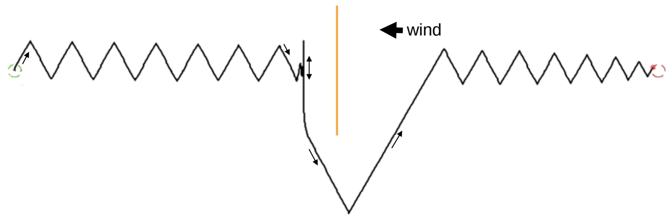
Small obstacles could be avoided easily on all simulated courses (Fig. 13(a), 14(a), 15(a)). When dealing with bigger obstacles the presented method starts to oscillate between two local optima v_t, which leads to the boat getting stuck in front of the obstacle (Fig. 14(b), 15(b)). This problem can be solved



(a)

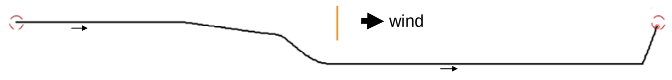


(b)

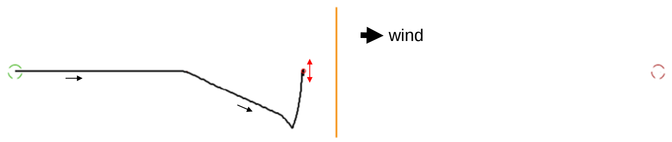


(c)

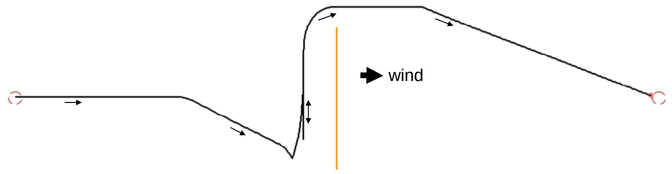
Fig. 14. Simulation results upwind



(a)



(b)



(c)

Fig. 15. Simulation results downwind

by detecting the oscillating behavior, i.e. the boat does not advance to the target significantly after multiple maneuvers and then gradually tightening the hysteresis condition (Fig. 14(c), 15(c)).

V. Conclusions and Future Work

The presented extension to short course routeing is able to avoid obstacles by adapting the underlying polar diagram by putting a penalty on directions where obstacles are located within a certain range. First simulations with simple static obstacles in various sizes are promising. It turned out that for large obstacles an adaptive hysteresis condition has to be implemented to be able to steer around the obstacle.

All experiments have been carried in computer simulation. Future experiments will include moving obstacles, more complex and non-convex obstacles as well as arrangements of multiple obstacles. The results will be verified with real world experiments.

References

[1] R. Stelzer and T. Proell, "Autonomous sailboat navigation for short course racing," *Robotics and Auotnomous System*, vol. 56, no. 7, pp. 604–614, July 2008.

[2] S. Showalter, "The legal status of autonomous underwater vehicles," *Mar. Techn. Soc. J.*, vol. 38, no. 1, p. 80–83, 2004.

[3] J. Larson, M. Bruch, R. Halterman, Rogers, and J. R. Webster, "Advances in autonomous obstacle avoidance for unmanned surface vehicles," in *Proceedings of AUVSI Unmanned Systems North America 2007*, vol. 582, Washington, DC, USA, 2007, p. 154pp.

[4] M. R. Benjamin, J. J. Leonard, J. A. Curcio, and P. M. Newman, "A method for protocol-based collision avoidance between autonomous marine surface craft," *J. Field Robot.*, vol. 23, no. 5, p. 333–346, 2006.

[5] T. Statheros, G. Howells, and K. McDonald-Maier, "Autonomous ship collision avoidance navigation concepts, technologies and techniques," *J. Navigation*, vol. 61, no. 1, p. 129–142, 2008.

[6] R. Smierzchalski, "Evolutionary-fuzzy system of safe ship steering in a collision situation at sea," in *International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, vol. 1, 2005, pp. 893–898.

[7] R. Stelzer and K. Jafarmadar, "Communication architecture for autonomous sailboats," International Robotic Sailing Conference (IRSC), Porto, Portugal, July 2009.

[8] Y. Briere, F. Bastianelli, M. Gagneul, and P. Cormerais, "Challenge microtransat," in *CETSIS'2005*, Nancy, France, 2005, vol. 5, Issue 2, doi:10.1051/j3ea:2006031.

[9] H. Klinck, R. Stelzer, K. Jafarmadar, and D. K. Mellinger, "Aas endurance: An autonomous acoustic sailboat for marine mammal research," in *International Robotic Sailing Conference (IRSC)*, Porto, Portugal, July 2009, pp. 43–48.

[10] R. Chatila and J.-C. Laumon, "Position referencing and consistent world modeling for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 1985, pp. 138–145.

[11] E. Gat, "On three-layer architectures," *Artificial Intelligence and Mobile Robots, AAAI Press*, 1998.

[12] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[13] K. H. Low, W. K. Leow, and M. H. Ang, "A hybrid mobile robot architecture with integrated planning and control," in *International joint conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, Bologna, Italy, 2002, pp. 219–226.

[14] A. Bonarini, G. Invernizzi, T. H. Labella, and M. Matteucci, "An architecture to coordinate fuzzy behaviors to control an autonomous robot," in *Fuzzy Sets and Systems*, vol. 134, 2003, pp. 101–115.

[15] J. Connell, "Sss: A hybrid architecture applied to robot navigation," in *IEEE International Conference on Robotics and Automation*, Nice, France, 1992, pp. 2719–2724.

[16] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, and J. O'Sullivan, "A layered architecture for office delivery robots," in *International Conference on Autonomous Agents (ICAA)*, 1997, pp. 235–242.

[17] R. Stelzer and K. Jafarmadar, "A layered system architecture to control an autonomous sailboat," in *Towards Autonomous Robotic Systems (TAROS 2007)*, Aberystwyth, UK, September, September 2007, pp. 153–159.

[18] P. J. Richards, A. Johnson, and A. Stanton, "America's cup downwind sails–vertical wings or horizontal parachutes?" *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 89, no. 14-15, pp. 1565–1577, 2001.

[19] M. Shamos and D. Hoey, "Geometric intersection problems," in *Proc. 17th IEEE Symp. Foundations of Computer Science (FOCS '76)*, vol. 134, 1976, pp. 208–215, doi:10.1109/SFCS.1976.16.

[20] C. Wylie, G. Romney, D. Evans, and A. Erdahl, "Halftone perspective drawings by computer," in *Proc. AFIPS FJCC*, vol. 31, 1967, p. 49.